waviiybyq

March 15, 2025

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

import warnings
warnings.simplefilter('ignore')
```

[2]:
```
df = pd.read_csv('/content/insurance.csv')
```

Columns

age: age of primary beneficiary

sex: insurance contractor gender, female, male

bmi: Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9

children: Number of children covered by health insurance / Number of dependents

smoker: Smoking

region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

charges: Individual medical costs billed by health insurance

Can you accurately predict insurance costs?

[3]:
```
df.head()
```

[3]:

	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

[4]:
```
df.describe()
```

[4]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

[5]:
```
df.shape
```

[5]:
```
(1338, 7)
```

[6]:
```
df.isnull().sum()
```

[6]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

[7]:
```
cat_col = df.select_dtypes('object')
```

[8]:
```
num_col = df.select_dtypes(['int64','float64'])
```

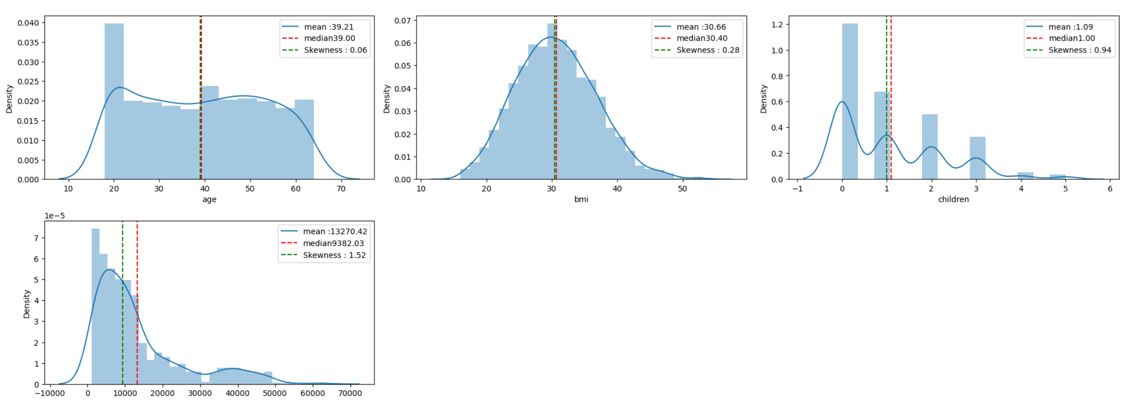
[9]:
```
for col in df.columns:
  print(col)
  print(df[col].unique())
  print('*'*75)
```

```
age
[19 18 28 33 32 31 46 37 60 25 62 23 56 27 52 30 34 59 63 55 22 26 35 24
 41 38 36 21 48 40 58 53 43 64 20 61 44 57 29 45 54 49 47 51 42 50 39]
***************************************************************************
sex
['female' 'male']
***************************************************************************
bmi
[27.9   33.77  33.    22.705 28.88  25.74  33.44  27.74  29.83  25.84
 26.22  26.29  34.4   39.82  42.13  24.6   30.78  23.845 40.3   35.3
 36.005 32.4   34.1   31.92  28.025 27.72  23.085 32.775 17.385 36.3
 35.6   26.315 28.6   28.31  36.4   20.425 32.965 20.8   36.67  39.9
 26.6   36.63  21.78  30.8   37.05  37.3   38.665 34.77  24.53  35.2
```

35.625	33.63	28.	34.43	28.69	36.955	31.825	31.68	22.88	37.335
27.36	33.66	24.7	25.935	22.42	28.9	39.1	36.19	23.98	24.75
28.5	28.1	32.01	27.4	34.01	29.59	35.53	39.805	26.885	38.285
37.62	41.23	34.8	22.895	31.16	27.2	26.98	39.49	24.795	31.3
38.28	19.95	19.3	31.6	25.46	30.115	29.92	27.5	28.4	30.875
27.94	35.09	29.7	35.72	32.205	28.595	49.06	27.17	23.37	37.1
23.75	28.975	31.35	33.915	28.785	28.3	37.4	17.765	34.7	26.505
22.04	35.9	25.555	28.05	25.175	31.9	36.	32.49	25.3	29.735
38.83	30.495	37.73	37.43	24.13	37.145	39.52	24.42	27.83	36.85
39.6	29.8	29.64	28.215	37.	33.155	18.905	41.47	30.3	15.96
33.345	37.7	27.835	29.2	26.41	30.69	41.895	30.9	32.2	32.11
31.57	26.2	30.59	32.8	18.05	39.33	32.23	24.035	36.08	22.3
26.4	31.8	26.73	23.1	23.21	33.7	33.25	24.64	33.88	38.06
41.91	31.635	36.195	17.8	24.51	22.22	38.39	29.07	22.135	26.8
30.02	35.86	20.9	17.29	34.21	25.365	40.15	24.415	25.2	26.84
24.32	42.35	19.8	32.395	30.2	29.37	34.2	27.455	27.55	20.615
24.3	31.79	21.56	28.12	40.565	27.645	31.2	26.62	48.07	36.765
33.4	45.54	28.82	22.99	27.7	25.41	34.39	22.61	37.51	38.
33.33	34.865	33.06	35.97	31.4	25.27	40.945	34.105	36.48	33.8
36.7	36.385	34.5	32.3	27.6	29.26	35.75	23.18	25.6	35.245
43.89	20.79	30.5	21.7	21.89	24.985	32.015	30.4	21.09	22.23
32.9	24.89	31.46	17.955	30.685	43.34	39.05	30.21	31.445	19.855
31.02	38.17	20.6	47.52	20.4	38.38	24.31	23.6	21.12	30.03
17.48	20.235	17.195	23.9	35.15	35.64	22.6	39.16	27.265	29.165
16.815	33.1	26.9	33.11	31.73	46.75	29.45	32.68	33.5	43.01
36.52	26.695	25.65	29.6	38.6	23.4	46.53	30.14	30.	38.095
28.38	28.7	33.82	24.09	32.67	25.1	32.56	41.325	39.5	34.3
31.065	21.47	25.08	43.4	25.7	27.93	39.2	26.03	30.25	28.93
35.7	35.31	31.	44.22	26.07	25.8	39.425	40.48	38.9	47.41
35.435	46.7	46.2	21.4	23.8	44.77	32.12	29.1	37.29	43.12
36.86	34.295	23.465	45.43	23.65	20.7	28.27	35.91	29.	19.57
31.13	21.85	40.26	33.725	29.48	32.6	37.525	23.655	37.8	19.
21.3	33.535	42.46	38.95	36.1	29.3	39.7	38.19	42.4	34.96
42.68	31.54	29.81	21.375	40.81	17.4	20.3	18.5	26.125	41.69
24.1	36.2	40.185	39.27	34.87	44.745	29.545	23.54	40.47	40.66
36.6	35.4	27.075	28.405	21.755	40.28	30.1	32.1	23.7	35.5
29.15	27.	37.905	22.77	22.8	34.58	27.1	19.475	26.7	34.32
24.4	41.14	22.515	41.8	26.18	42.24	26.51	35.815	41.42	36.575
42.94	21.01	24.225	17.67	31.5	31.1	32.78	32.45	50.38	47.6
25.4	29.9	43.7	24.86	28.8	29.5	29.04	38.94	44.	20.045
40.92	35.1	29.355	32.585	32.34	39.8	24.605	33.99	28.2	25.
33.2	23.2	20.1	32.5	37.18	46.09	39.93	35.8	31.255	18.335
42.9	26.79	39.615	25.9	25.745	28.16	23.56	40.5	35.42	39.995
34.675	20.52	23.275	36.29	32.7	19.19	20.13	23.32	45.32	34.6
18.715	21.565	23.	37.07	52.58	42.655	21.66	32.	18.3	47.74
22.1	19.095	31.24	29.925	20.35	25.85	42.75	18.6	23.87	45.9
21.5	30.305	44.88	41.1	40.37	28.49	33.55	40.375	27.28	17.86
33.3	39.14	21.945	24.97	23.94	34.485	21.8	23.3	36.96	21.28

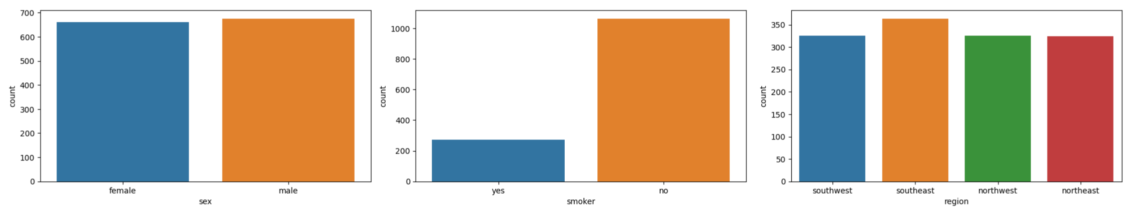
```
 29.4    27.3    37.9    37.715  23.76   25.52   27.61   27.06   39.4    34.9
 22.     30.36   27.8    53.13   39.71   32.87   44.7    30.97 ]
***************************************************************************
children
[0 1 3 2 5 4]
***************************************************************************
smoker
['yes' 'no']
***************************************************************************
region
['southwest' 'southeast' 'northwest' 'northeast']
***************************************************************************
charges
[16884.924   1725.5523  4449.462  …  1629.8335  2007.945  29141.3603]
***************************************************************************
```

[10]:
```python
a = len(df.columns)
b = 3
c = 1

fig = plt.figure(figsize=(20,25))

for col in num_col:
  plt.subplot(a,b,c)
  plt.xlabel(col)
  sns.distplot(x=df[col])
  plt.axvline(x=np.mean(df[col]),c='r',ls='--')
  plt.axvline(x=np.median(df[col]),c='g',ls='--')
  plt.legend(('mean :%.2f'%(np.mean(df[col])),'median%.2f'%(np.
 ↪median(df[col])),'Skewness : %.2f'%(df[col].skew())))
  c = c+1

plt.tight_layout()
plt.show()
```

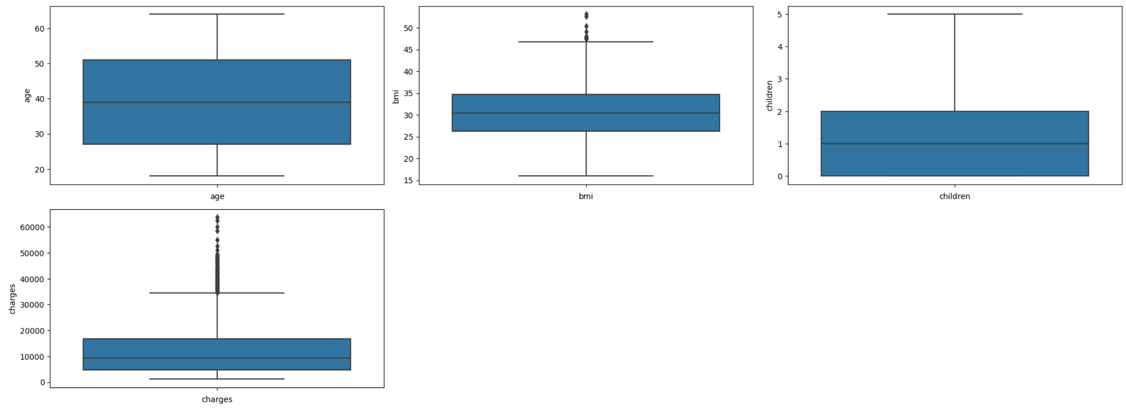


[11]:
```
#charges and children are positively skewed
#the other variable are relatively normally distributed
```

[12]:
```
a = len(df.columns)
b = 3
c = 1

fig = plt.figure(figsize=(20,25))

for col in cat_col:
  plt.subplot(a,b,c)
  plt.xlabel(col)
  sns.countplot(x=df[col])
  c = c+1

plt.tight_layout()
plt.show()
```

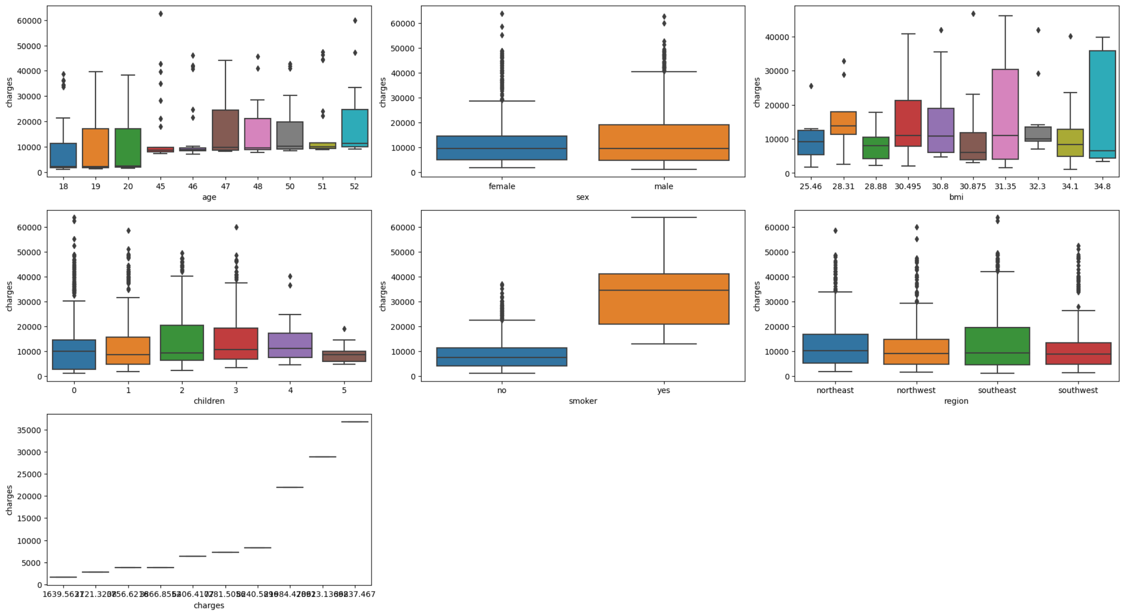


[13]:
```
a = len(df.columns)
b = 3
c = 1

fig = plt.figure(figsize=(20,25))

for col in num_col:
  plt.subplot(a,b,c)
  plt.xlabel(col)
  sns.boxplot(y=df[col])
  c = c+1

plt.tight_layout()
plt.show()
```



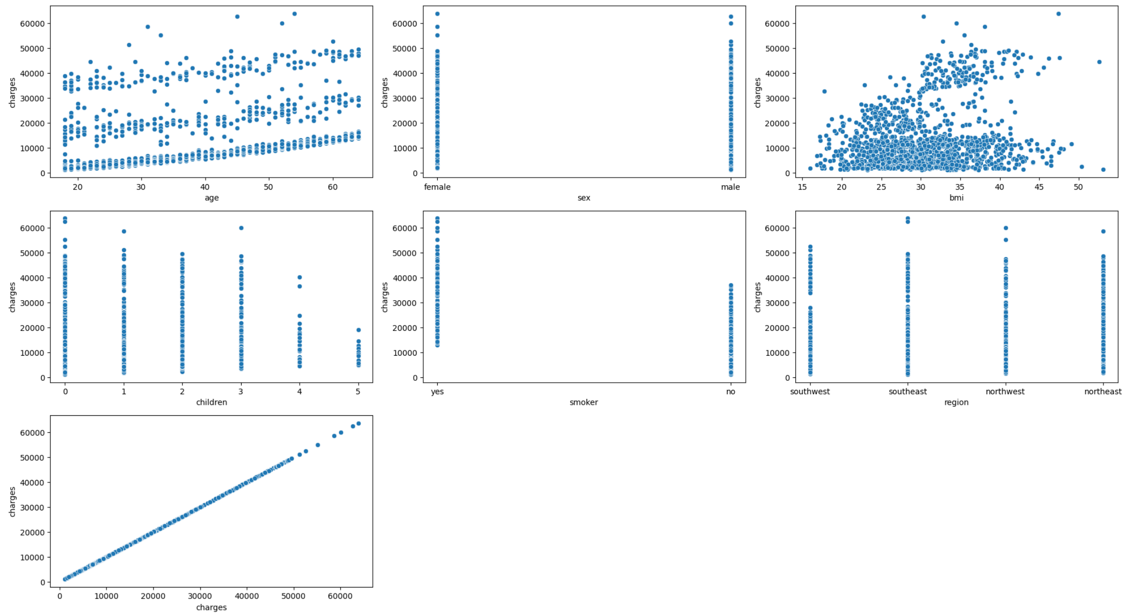
[14]:
```
#a lot of outliers in charges
```

[15]:
```
a = len(df.columns)
b = 3
c = 1

fig = plt.figure(figsize=(20,25))

for col in df.columns:
  plt.subplot(a,b,c)
  plt.xlabel(col)
  sns.boxplot(x=df[col],y=df['charges'],order=df[col].value_counts().
 ↪sort_values(ascending=False).index[:10].sort_values(ascending=True))
  c = c+1

plt.tight_layout()
plt.show()
```

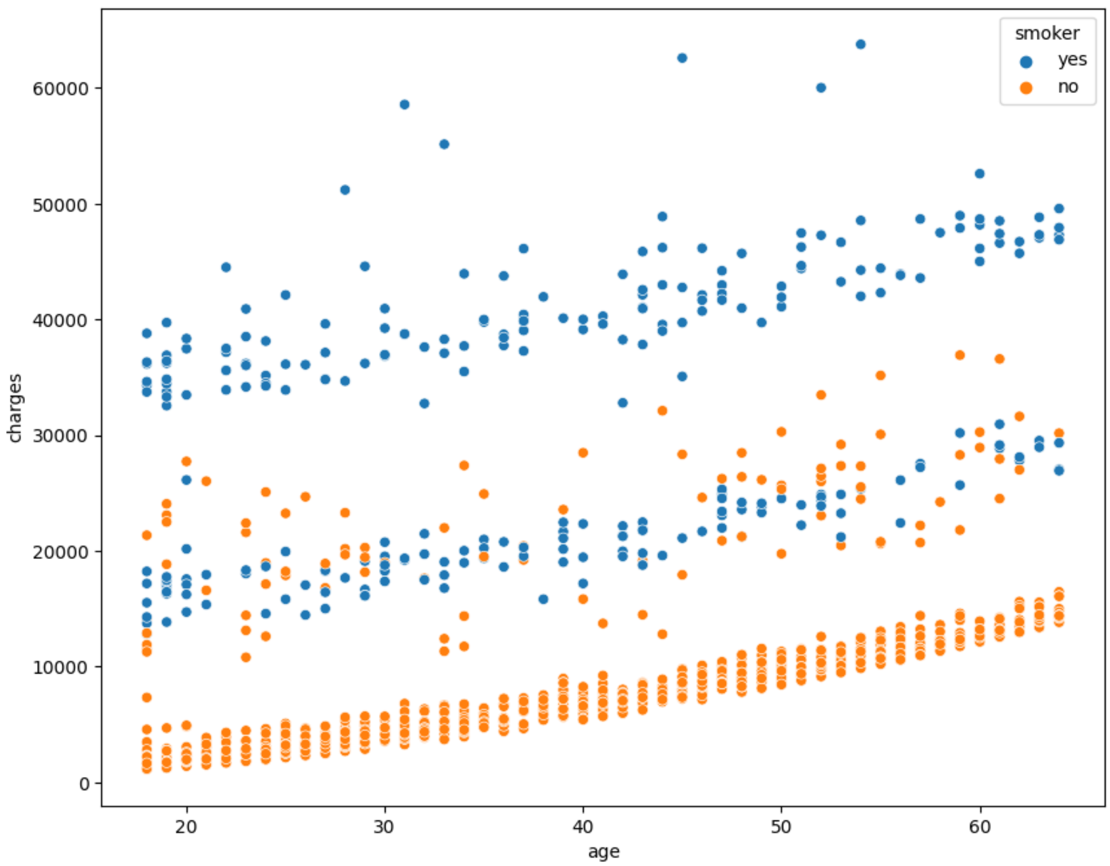


[16]:
```
#charges go up with age
#higher for male and smokers
```

[17]:
```
a = len(df.columns)
b = 3
c = 1

fig = plt.figure(figsize=(20,25))

for col in df.columns:
  plt.subplot(a,b,c)
  plt.xlabel(col)
  sns.scatterplot(x=col,y='charges',data=df)
  c = c+1

plt.tight_layout()
plt.show()
```

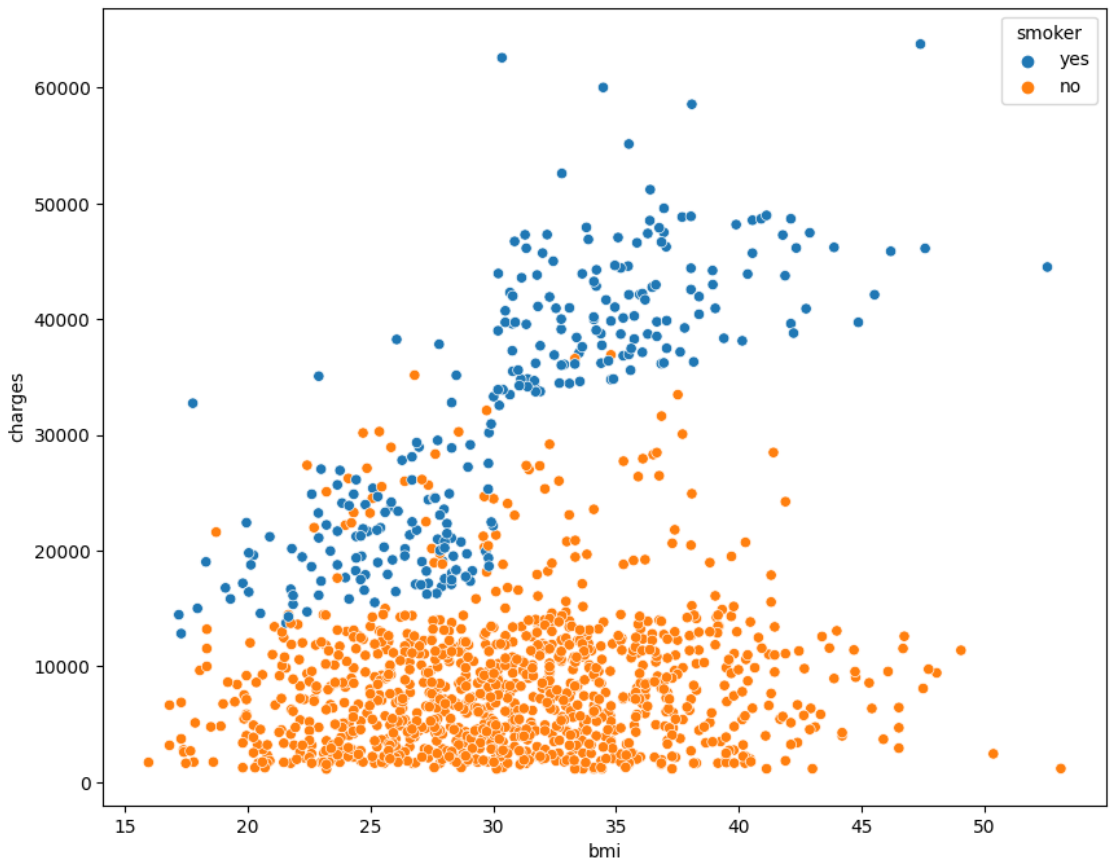


[18]:
```
plt.figure(figsize=(10,8))
sns.scatterplot(data=df,y='charges',x='age',hue='smoker')
plt.show()
```



[19]:
```
#non smokers have the least ammount of charges
#smokers are in the upper two clusters
#charges go up with age
```

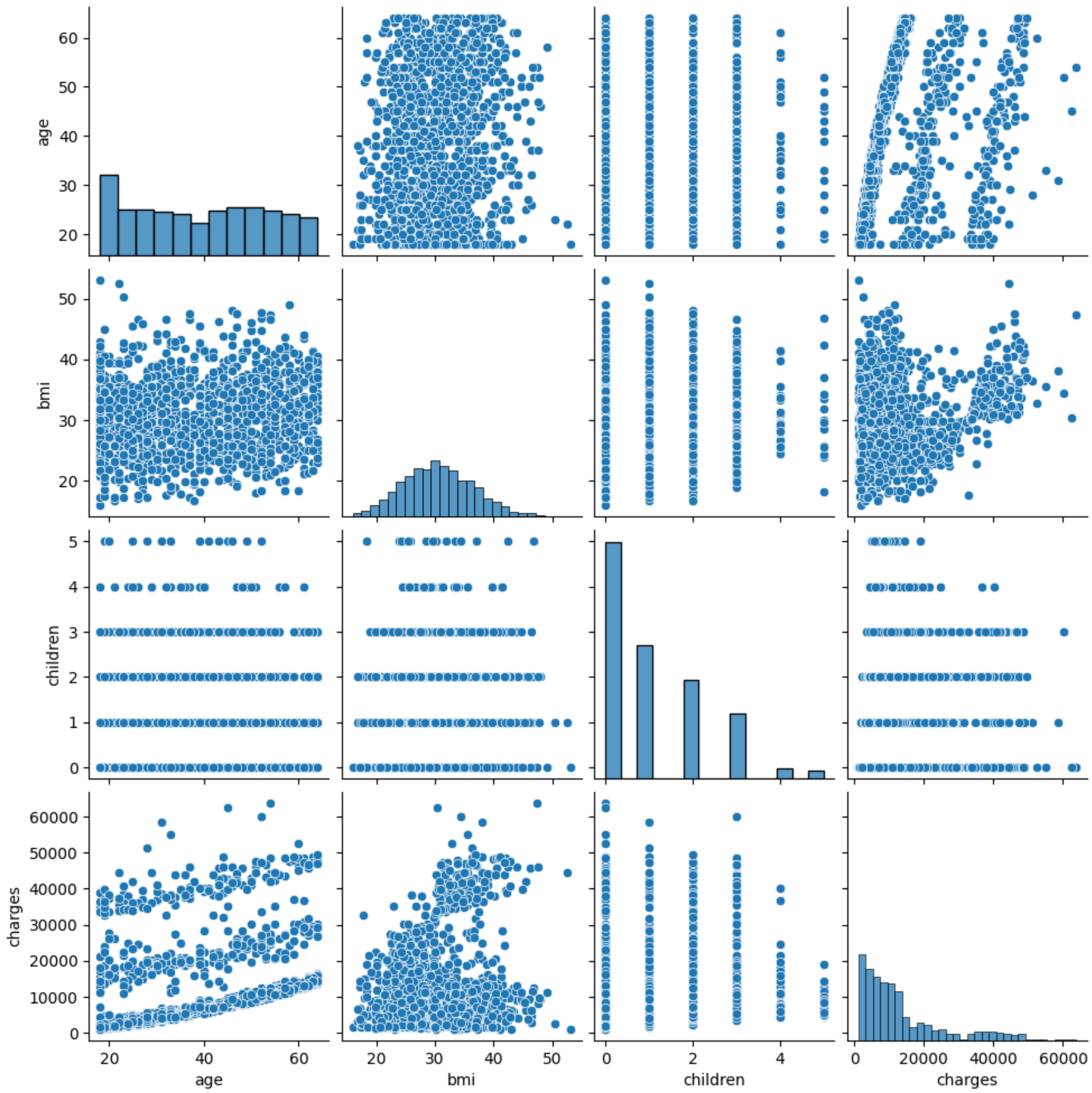
[20]:
```
plt.figure(figsize=(10,8))
sns.scatterplot(data=df,y='charges',x='bmi',hue='smoker')
plt.show()
```



[21]:
```
#charges higher if you smoke no matter your bmi
```

[22]:
```
sns.pairplot(df)
```

[22]: <seaborn.axisgrid.PairGrid at 0x7f99429ab7c0>



[23]:
```
df.head()
```

[23]:

	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

[24]:
```
df['sex'] = df['sex'].replace({'female':0,'male':1})
df['smoker'] = df['smoker'].replace({'no':0,'yes':1})
df['smoker'] = df['smoker'].replace({'no':0,'yes':1})
```

```
df['region'] = df['region'].replace({'northeast':0,'northwest':1,'southeast':
↪2,'southwest':3})
```

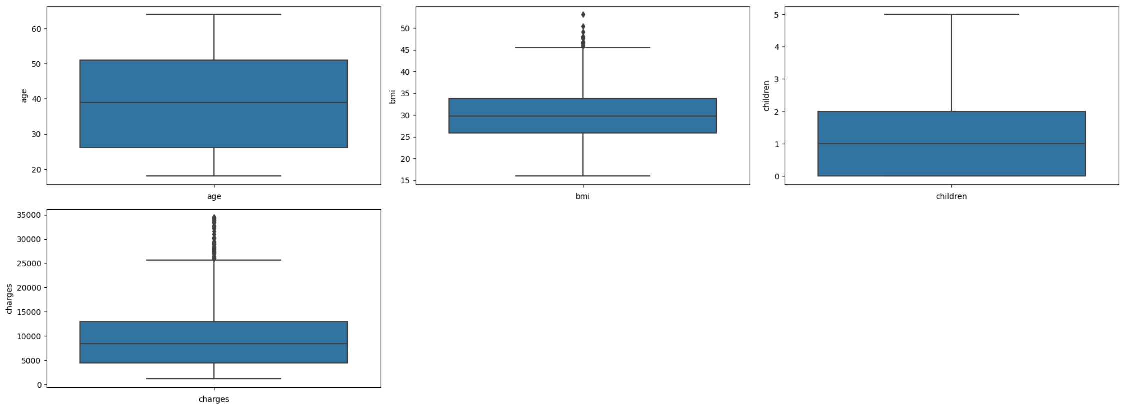
[25]:

```
# check outliers

for col in df.columns:
  q1 = df[col].quantile(0.25)
  q3 = df[col].quantile(0.75)
  iqr=q3-q1
  lower_tail = q1 - 1.5 * iqr
  upper_tail = q3 + 1.5 * iqr
  data = df[(df[col] < upper_tail) & (df[col] > lower_tail)]

print(df.shape)
print('*'*10)
print(data.shape)
```

```
(1338, 7)
**********
(1199, 7)
```

[26]:

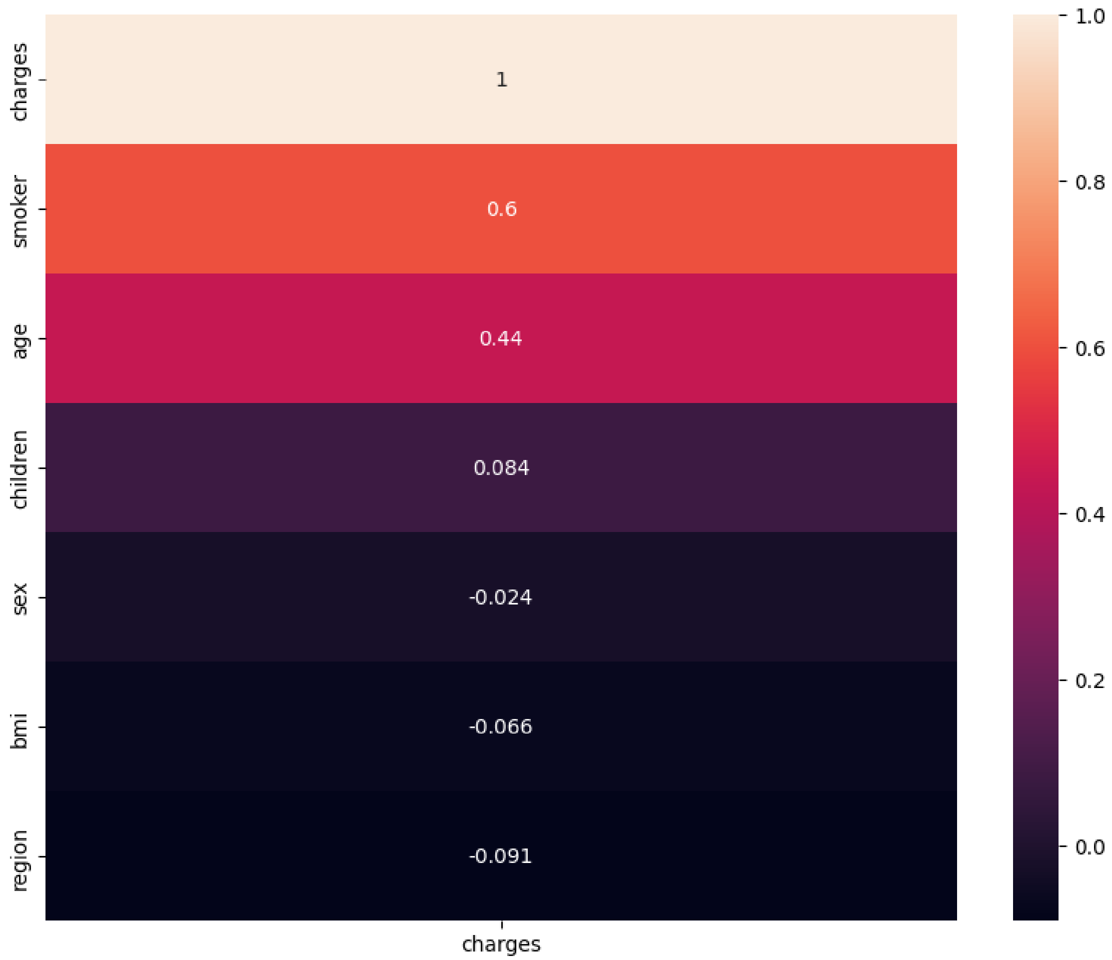
```
a = len(df.columns)
b = 3
c = 1

fig = plt.figure(figsize=(20,25))

for col in num_col:
  plt.subplot(a,b,c)
  plt.xlabel(col)
  sns.boxplot(y=data[col])
  c = c+1

plt.tight_layout()
plt.show()
#still a few but better
```



[27]:
```python
# check multico

plt.figure(figsize=(10,8))
sns.heatmap(data.corr(),annot=True)
plt.show()
```

[28]:

```
plt.figure(figsize=(10,8))
sns.heatmap(data.corr()[['charges']].
 ↪sort_values(by='charges',ascending=False),annot=True)
plt.show()
```



[29]:

```
#as expected smoker and charges are highly positively correlated
```

[30]:

```
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()

data_scaled = pd.DataFrame(ss.fit_transform(data),columns=data.
 ↪columns,index=data.index)
```

[31]:
```python
# OLS

from statsmodels.formula.api import ols
from statsmodels.stats import diagnostic
import statsmodels.api as sm


X = data_scaled.drop('charges',axis=1)
y = data_scaled['charges']
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()
print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.604
Model:                            OLS   Adj. R-squared:                  0.602
Method:                 Least Squares   F-statistic:                     303.4
Date:                Sun, 09 Apr 2023   Prob (F-statistic):          7.29e-236
Time:                        22:40:47   Log-Likelihood:                -1145.6
No. Observations:                1199   AIC:                             2305.
Df Residuals:                    1192   BIC:                             2341.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.689e-17      0.018   1.48e-15      1.000      -0.036       0.036
age            0.4717      0.018     25.651      0.000       0.436       0.508
sex           -0.0252      0.018     -1.381      0.168      -0.061       0.011
bmi            0.0533      0.019      2.783      0.005       0.016       0.091
children       0.0701      0.018      3.842      0.000       0.034       0.106
smoker         0.6449      0.019     34.187      0.000       0.608       0.682
region        -0.0718      0.018     -3.891      0.000      -0.108      -0.036
==============================================================================
Omnibus:                      751.833   Durbin-Watson:                   2.061
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5270.438
Skew:                           3.013   Prob(JB):                         0.00
Kurtosis:                      11.318   Cond. No.                         1.39
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[32]:
```
# R2 = 60%, pouvoir explicatif décent
#p value sex > 0.5 on ne rejette pas H0 hypothèse de non significativité au␣
↪seuil 5%
#poru toutes les autres variables p-value < 0.05 on rejette  H0 hypothèse de␣
↪non significativité au seuil 5%
```

[33]:
```
data_signi = data_scaled.drop('sex',axis=1)
```

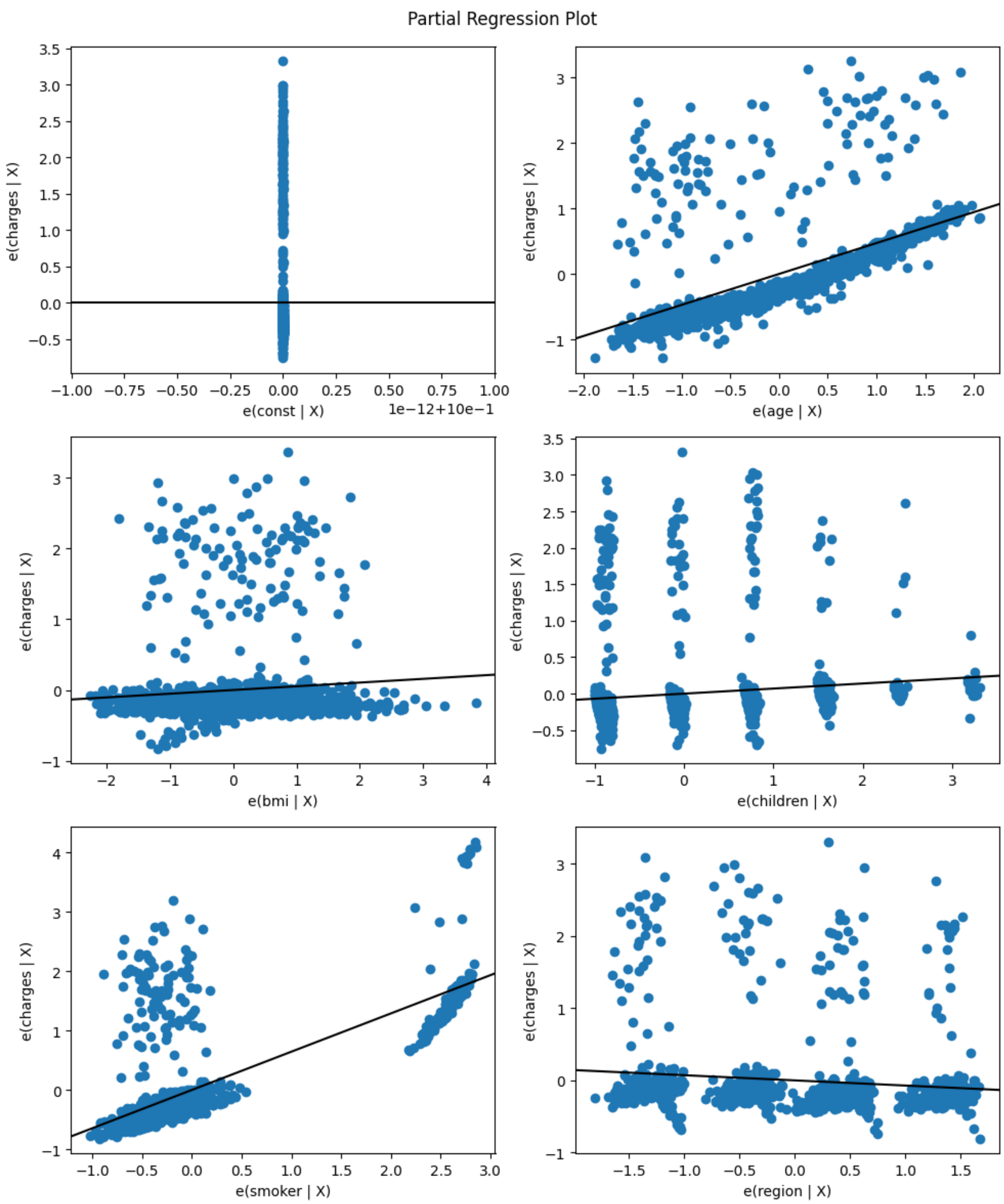
[34]:
```
X = data_signi.drop('charges',axis=1)
y = data_signi['charges']
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

y_pred = model.predict()
residuals = y - y_pred

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.604
Model:                            OLS   Adj. R-squared:                  0.602
Method:                 Least Squares   F-statistic:                     363.4
Date:                Sun, 09 Apr 2023   Prob (F-statistic):          9.40e-237
Time:                        22:40:47   Log-Likelihood:                -1146.5
No. Observations:                1199   AIC:                             2305.
Df Residuals:                    1193   BIC:                             2336.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       2.689e-17      0.018   1.48e-15      1.000      -0.036       0.036
age            0.4723      0.018     25.683      0.000       0.436       0.508
bmi            0.0524      0.019      2.735      0.006       0.015       0.090
children       0.0697      0.018      3.821      0.000       0.034       0.106
smoker         0.6444      0.019     34.153      0.000       0.607       0.681
region        -0.0715      0.018     -3.878      0.000      -0.108      -0.035
==============================================================================
Omnibus:                      749.383   Durbin-Watson:                   2.060
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5221.487
Skew:                           3.002   Prob(JB):                         0.00
Kurtosis:                      11.274   Cond. No.                         1.38
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[35]:
```python
# Check assumptions
# linéarité

fig = plt.figure(figsize=(10,12))
fig = sm.graphics.plot_partregress_grid(model, fig=fig)
```

```
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
eval_env: 1
```



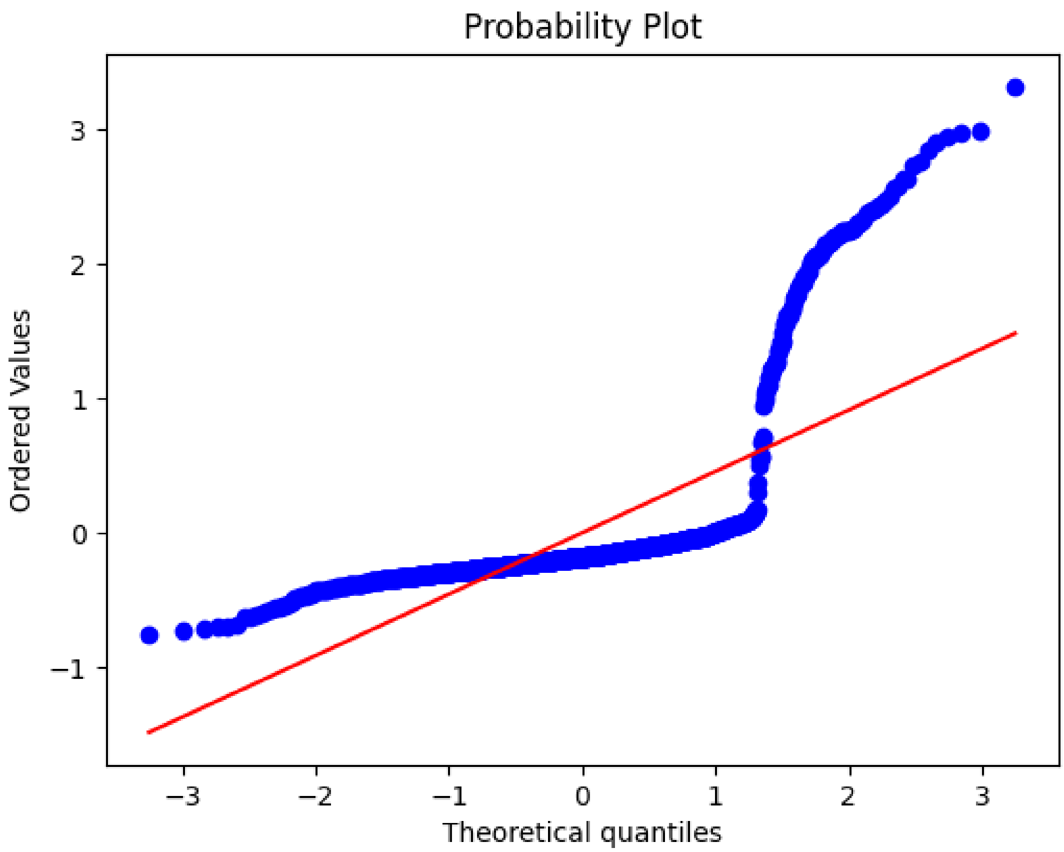
[36]:
```
#linear relationship between depedent variable and independent ones
```

[37]:
```
mean_residuals = np.mean(residuals)
print("Mean of Residuals {}".format(mean_residuals))
#very close to zero
```

```
Mean of Residuals 1.4815319761469978e-18
```

[38]:

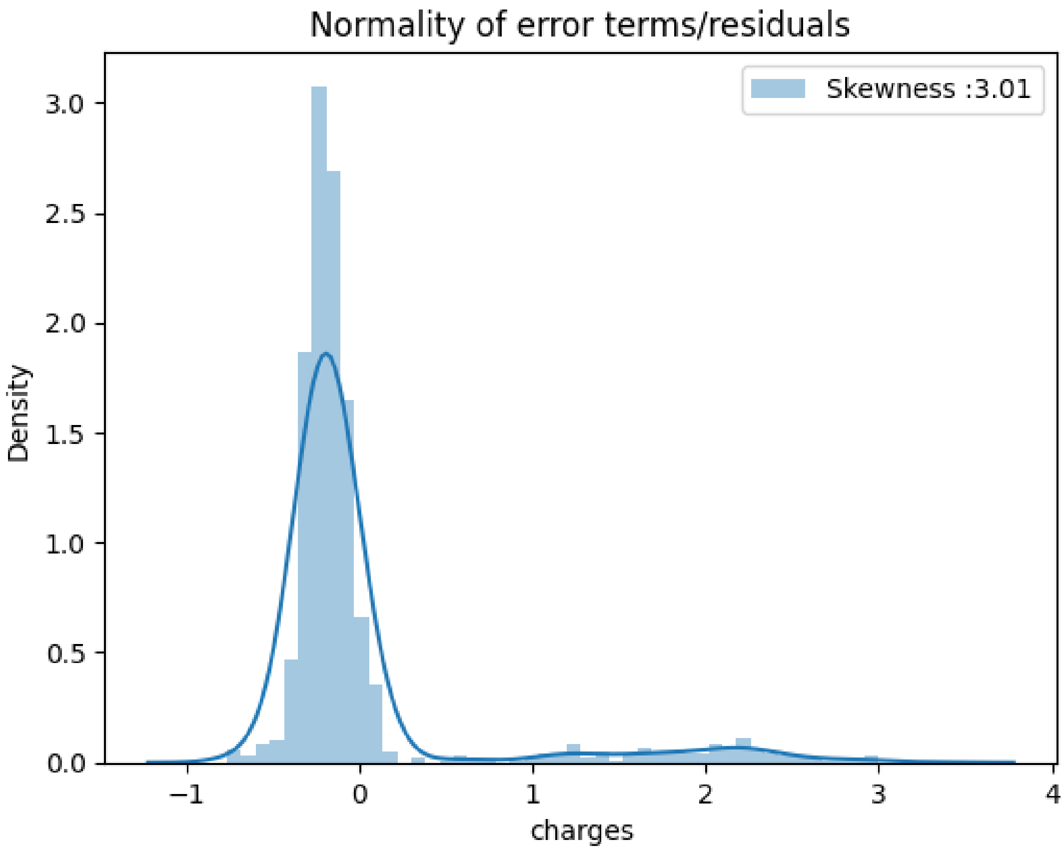
```
import pylab
import scipy.stats as stats

stats.probplot(residuals, dist="norm", plot=pylab)
plt.show()
```



[39]:

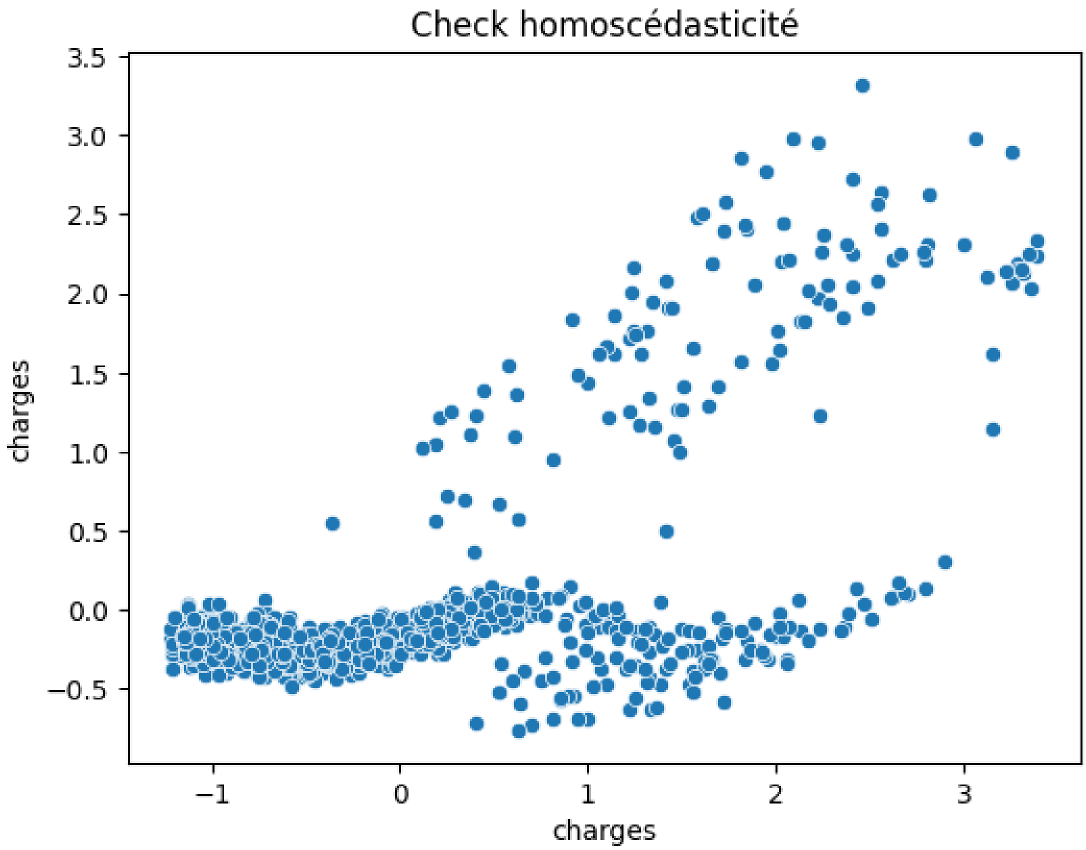
```
sns.distplot(residuals,kde=True,label='Skewness :%.2f'%(residuals.skew()))
plt.legend()
plt.title('Normality of error terms/residuals')
```

[39]: Text(0.5, 1.0, 'Normality of error terms/residuals')



[40]:
```
#prob(jb) < 0.5 on rejette H0 hypothese de distribution normale des résidus au␣
↪seuil 5%
```

[41]:
```
sns.scatterplot(x=y,y=residuals)
plt.title('Check homoscédasticité')
plt.show()
```



[42]:
```
import statsmodels.stats.api as sms
from statsmodels.compat import lzip


name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]
test = sms.het_breuschpagan(model.resid, model.model.exog)
lzip(name, test)
```

[42]:
```
[('Lagrange multiplier statistic', 15.798373953724221),
 ('p-value', 0.007443933940098722),
 ('f-value', 3.1858408088522863),
 ('f p-value', 0.007304899390989418)]
```

[43]:
```
name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]
test = sm.stats.diagnostic.het_white(model.resid, model.model.exog)
lzip(name, test)
```

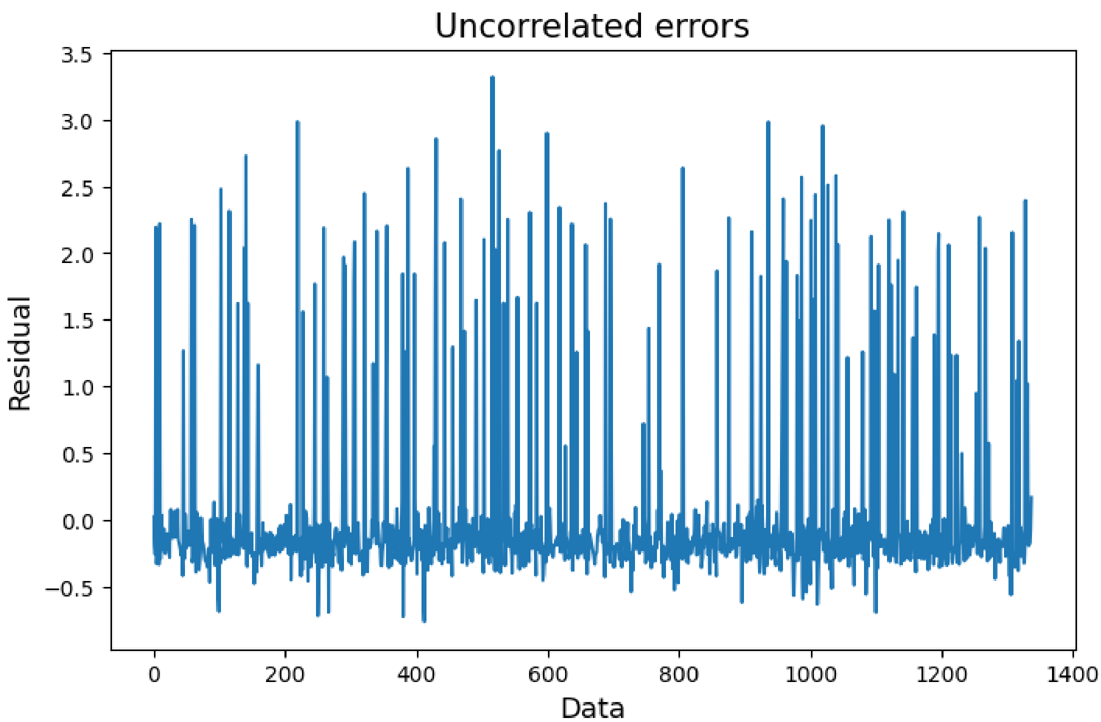
[43]:
```
[('Lagrange multiplier statistic', 63.179332053793125),
 ('p-value', 1.2029182219740895e-06),
 ('f-value', 3.451639793125934),
```

```
('f p-value', 7.991054086746958e-07)]
```

[44]:
```
# p-value < 0,05 on rejette H0: hypothese d'homoscedasticité
```

[45]:
```
residuals_info = pd.DataFrame({'y_true': y, 'y_pred': y_pred, 'error':␣
↪residuals}, columns=['y_true', 'y_pred', 'error'])
fig, ax = plt.subplots(figsize=(8,5))
ax = residuals_info.error.plot()
ax.set_title('Uncorrelated errors', fontsize=15)
ax.set_xlabel("Data", fontsize=13)
ax.set_ylabel("Residual", fontsize=13)
```

[45]:
```
Text(0, 0.5, 'Residual')
```



[46]:
```
# DW = 2 on conclut donc a l'abscence d'autocorellation des erreurs
```

[47]:
```
import statsmodels.api as sm


def vif_cal(input_data, dependent_col):
    vif_df = pd.DataFrame( columns = ['Var', 'Vif'])
    x_vars=input_data.drop([dependent_col], axis=1)
    xvar_names=x_vars.columns
```

```python
    for i in range(0,xvar_names.shape[0]):
        y=x_vars[xvar_names[i]]
        x=x_vars[xvar_names.drop(xvar_names[i])]
        rsq=sm.OLS(y,x).fit().rsquared
        vif=round(1/(1-rsq),2)
        vif_df.loc[i] = [xvar_names[i], vif]
    return vif_df.sort_values(by = 'Vif', axis=0, ascending=False,␣
↪inplace=False)
```

[48]:
```python
vif_cal(input_data=data, dependent_col='charges')
```

[48]:

	Var	Vif
2	bmi	10.05
0	age	7.60
5	region	2.85
1	sex	1.92
3	children	1.77
4	smoker	1.10

[49]:
```python
#vif > 5 pour bmi et age, signes de multicolinéarité
```

[50]:
```python
data_log = data.copy()
data_log['charges'] = np.log(data_log['charges'])
```

[51]:
```python
#data_log.replace([np.inf, -np.inf], np.nan, inplace=True)
#data_log.dropna(inplace=True)
```

[52]:
```python
#data_log.head()
```

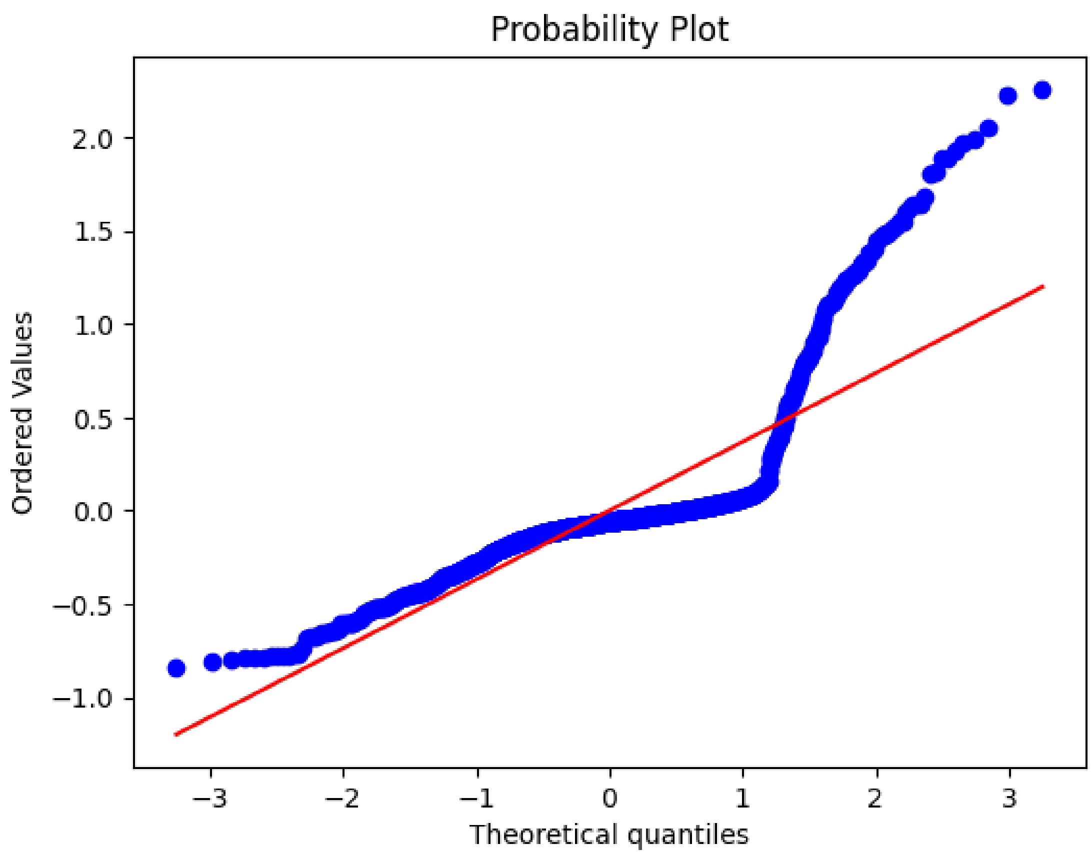
[53]:
```python
X = data_log.drop('charges',axis=1)
y = data_log['charges']
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

y_pred = model.predict()
residuals = y - y_pred

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.709
Model:                            OLS   Adj. R-squared:                  0.708
Method:                 Least Squares   F-statistic:                     484.7
Date:                Sun, 09 Apr 2023   Prob (F-statistic):          1.53e-315
```

```
Time:                        22:40:54   Log-Likelihood:                -687.27
No. Observations:                1199   AIC:                             1389.
Df Residuals:                    1192   BIC:                             1424.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          7.1926      0.074     96.650      0.000       7.047       7.339
age            0.0373      0.001     41.803      0.000       0.036       0.039
sex           -0.0874      0.025     -3.510      0.000      -0.136      -0.039
bmi            0.0045      0.002      2.048      0.041       0.000       0.009
children       0.1084      0.010     10.567      0.000       0.088       0.128
smoker         1.3134      0.040     32.562      0.000       1.234       1.393
region        -0.0582      0.011     -5.130      0.000      -0.080      -0.036
==============================================================================
Omnibus:                      574.783   Durbin-Watson:                   2.015
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3010.397
Skew:                           2.240   Prob(JB):                         0.00
Kurtosis:                       9.339   Cond. No.                         310.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[54]:
```
#better R2
#p-values de toutes les variables <0.05, on rejette  H0 hypothese de non␣
 ↪significativité
#DW = 2 on suppose absence d'autocorrelation des erreus
#Prob(JB) < 0,05 on rejette H0: distribution normale des résidus
```

[55]:
```
stats.probplot(residuals, dist="norm", plot=pylab)
plt.show()
```



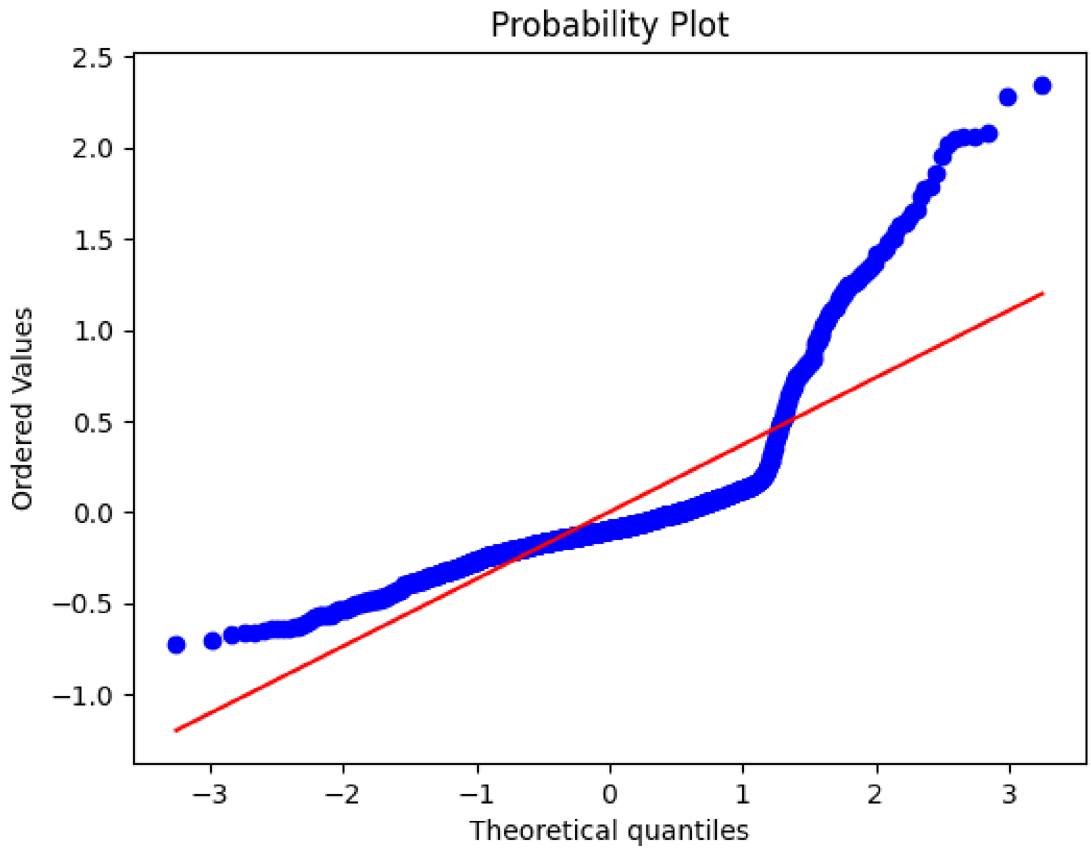
[56]:
```python
data_log_2 = data.copy()
for col in data_log_2:
  data_log_2[col] = np.log1p(data_log_2[col])
```

[57]:
```python
X = data_log_2.drop('charges',axis=1)
y = data_log_2['charges']
X = sm.add_constant(X)

model = sm.OLS(y, X).fit()

y_pred = model.predict()
residuals = y - y_pred

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                charges   R-squared:                       0.709
Model:                            OLS   Adj. R-squared:                  0.707
```

```
Method:                 Least Squares   F-statistic:                     483.5
Date:                Sun, 09 Apr 2023   Prob (F-statistic):          4.27e-315
Time:                        22:40:55   Log-Likelihood:                -688.07
No. Observations:                1199   AIC:                             1390.
Df Residuals:                    1192   BIC:                             1426.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.1646      0.247     12.791      0.000       2.679       3.650
age            1.3733      0.033     41.448      0.000       1.308       1.438
sex           -0.1243      0.036     -3.458      0.001      -0.195      -0.054
bmi            0.1963      0.067      2.917      0.004       0.064       0.328
children       0.1884      0.022      8.426      0.000       0.145       0.232
smoker         1.9071      0.058     32.762      0.000       1.793       2.021
region        -0.1326      0.024     -5.432      0.000      -0.181      -0.085
==============================================================================
Omnibus:                      611.743   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3389.212
Skew:                           2.396   Prob(JB):                         0.00
Kurtosis:                       9.699   Cond. No.                         107.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

[58]:
```python
stats.probplot(residuals, dist="norm", plot=pylab)
plt.show()
```



[59]:
```
#the same assumptions are still being violated
```

[60]:
```
#other regression models with less regarding assumptions

from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

#using the same df than the most accurate OLS regression

X = data_log_2.drop('charges',axis=1)
y = data_log_2['charges']

X_train, X_test,y_train, y_test = train_test_split(X,y,test_size=0.3)
```

[61]:
```
X_scaled = ss.fit_transform(X_train)
X_test_scaled = ss.transform(X_test)
```

[62]:
```
dtr  = DecisionTreeRegressor()
dtr.fit(X_scaled, y_train)
y_pred = dtr.predict(X_test_scaled)
r2_score(y_test,y_pred)
```

[62]: 0.4883614600077617

[63]:
```
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(dtr, X_train, y_train, cv=10, scoring='r2')
cv_scores.mean()
```

[63]: 0.5763590984641845

[63]:
```

```