

# Insurance Premium

September 3, 2023

Cissé NIANG

Prédiction du coût de l'assurance médicale

```
[1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import kurtosis, skew, stats
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn import linear_model
import warnings
warnings.filterwarnings("ignore")
```

```
[2]: df=pd.read_csv('/Users/cisseniang/Documents/Data/Données ML/Insurance.csv')
```

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

```
[3]:
```

	age	sex	bmi	children	smoker	region	expenses
0	19	female	27.9	0	yes	southwest	16884.92
1	18	male	33.8	1	no	southeast	1725.55
2	28	male	33.0	3	no	southeast	4449.46
3	33	male	22.7	0	no	northwest	21984.47
4	32	male	28.9	0	no	northwest	3866.86

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         1338 non-null   int64
1   sex         1338 non-null   object
2   bmi         1338 non-null   float64
```

```

3  children  1338 non-null   int64
4  smoker    1338 non-null   object
5  region    1338 non-null   object
6  expenses  1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB

```

## 0.1 Vérifier les données manquantes

```
[5]: (df.isnull().sum()/df.shape[0]).sort_values(ascending = False)
```

```

[5]: age          0.0
     sex          0.0
     bmi          0.0
     children     0.0
     smoker       0.0
     region       0.0
     expenses     0.0
     dtype: float64

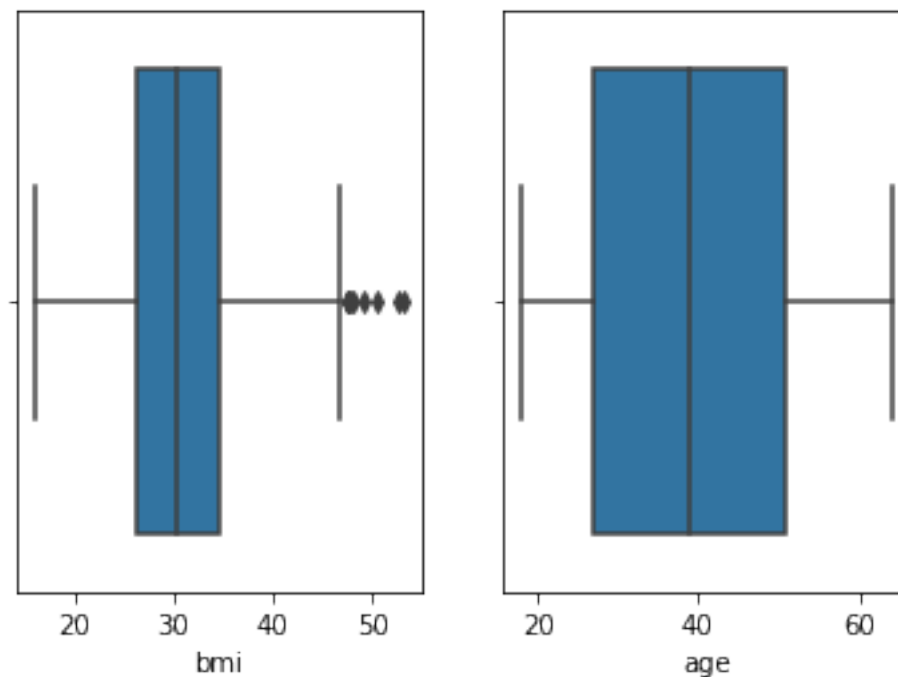
```

## 0.2 Visualisations et statistiques descriptives des données

```

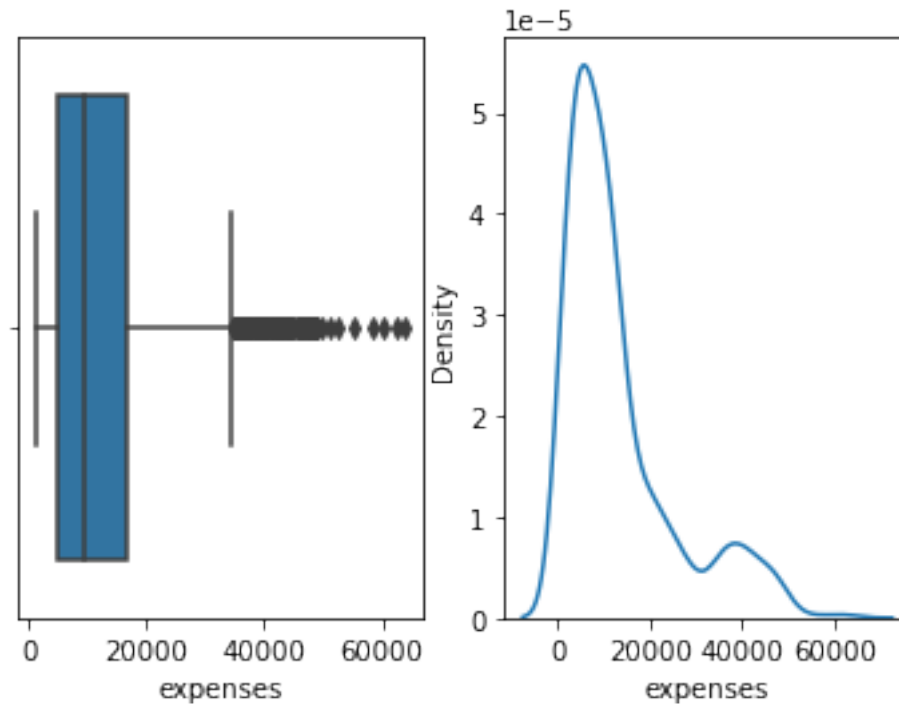
[6]: f, axes = plt.subplots(1, 2)
     plt.figure(figsize=(16,8))
     sns.boxplot(df['bmi'], ax=axes[0])
     sns.boxplot(df['age'], ax=axes[1])
     plt.show()

```



<Figure size 1152x576 with 0 Axes>

```
[7]: f, axes = plt.subplots(1, 2)
sns.boxplot(df['expenses'], ax=axes[0])
sns.kdeplot(df['expenses'], ax=axes[1])
plt.show()
```



Le diagramme de la densité nous montre que les données relatives aux charges (expenses) sont asymétriques à droite, de même que le diagramme en boîte nous montre que La distribution est positivement asymétrique, car la portion droite de la boîte et la moustache droite sont plus longues qu'à gauche de la médiane

```
[8]: df[['age', 'bmi', 'expenses']].describe().T
```

```
[8]:
```

	count	mean	std	min	25%	50%	\
age	1338.0	39.207025	14.049960	18.00	27.0000	39.00	
bmi	1338.0	30.665471	6.098382	16.00	26.3000	30.40	
expenses	1338.0	13270.422414	12110.011240	1121.87	4740.2875	9382.03	
	75%	max					
age	51.000	64.00					
bmi	34.700	53.10					

```
expenses 16639.915 63770.43
```

Pour explorer le comportement de certaines variables par rapport aux autres, nous allons recourir à d'autres indicateurs statistiques tels que la matrice de corrélation et la matrice de covariance.

En effet, la matrice de covariance mesure la covariation entre deux variables aléatoires. Elle donne une idée de la manière dont les variables évoluent conjointement. Si la covariance est positive, cela signifie que les deux variables tendent à augmenter ensemble, tandis qu'une covariance négative indique que lorsque l'une des variables augmente, l'autre tend à diminuer. Une covariance nulle signifie qu'il n'y a pas de relation linéaire entre les variables.

$$\text{Cov}(X, Y) = \Sigma[(x_i - \bar{x}) * (y_i - \bar{y})] / (n - 1)$$

```
[9]: df.cov()
```

```
[9]:
```

	age	bmi	children	expenses
age	197.401387	9.368560	0.719303	5.087480e+04
bmi	9.368560	37.190265	0.092958	1.466515e+04
children	0.719303	0.092958	1.453213	9.926742e+02
expenses	50874.802133	14665.149703	992.674243	1.466524e+08

La matrice de corrélation est une version normalisée de la matrice de covariance. Elle mesure la force et la direction de la relation linéaire entre deux variables, tout en éliminant l'effet de l'échelle. La corrélation est comprise entre -1 et 1. Une corrélation de 1 indique une relation linéaire positive parfaite, -1 indique une relation linéaire négative parfaite, et 0 indique aucune corrélation linéaire.

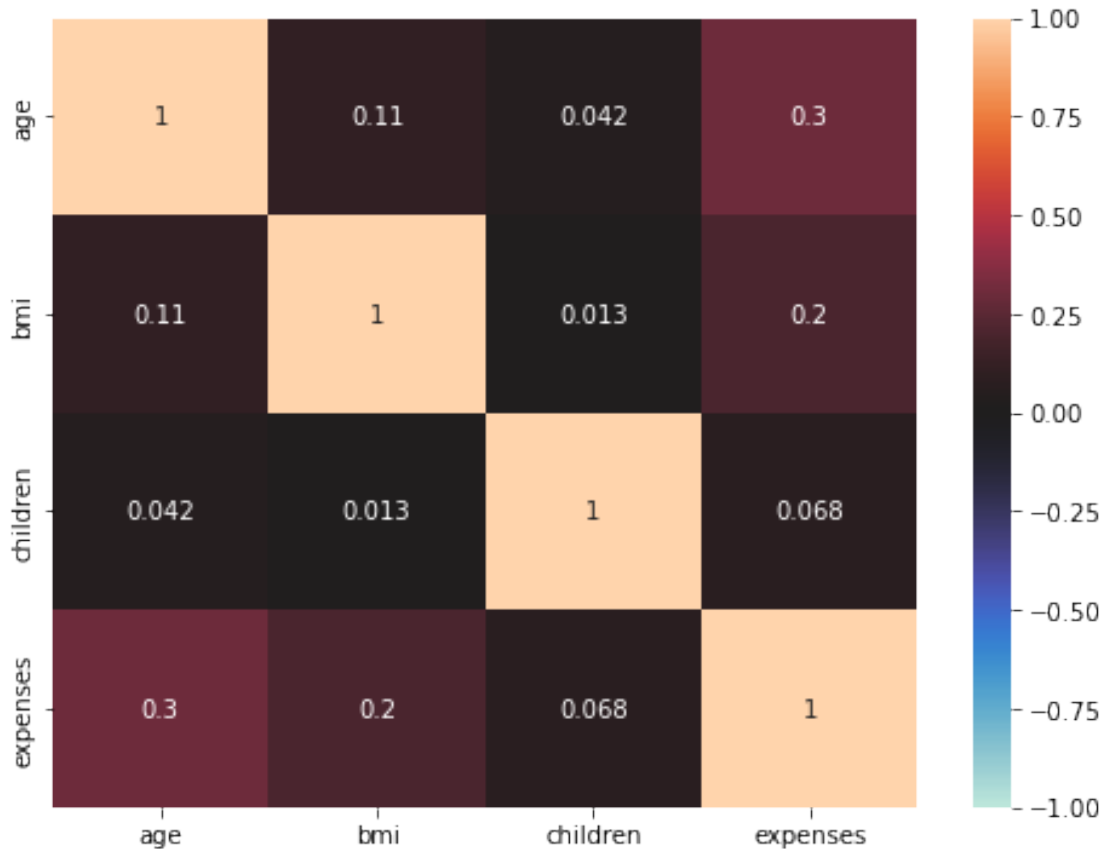
$$\text{Corr}(X, Y) = \text{Cov}(X, Y) / (\sigma_X * \sigma_Y)$$

```
[10]: df.corr()
```

```
[10]:
```

	age	bmi	children	expenses
age	1.000000	0.109341	0.042469	0.299008
bmi	0.109341	1.000000	0.012645	0.198576
children	0.042469	0.012645	1.000000	0.067998
expenses	0.299008	0.198576	0.067998	1.000000

```
[11]: plt.figure(figsize=(8,6))
ax = sns.heatmap(df.corr(),vmin=-1,vmax=1,center=0,annot=True)
```



Cette matrice de corrélation nous montre que :

1- il y a une corrélation faible entre l'âge et l'indice de masse corporelle (bmi), une corrélation encore plus faible avec le nombre d'enfants (children), et une corrélation modérée avec les dépenses médicales (expenses). La corrélation positive avec les dépenses médicales suggère que les personnes plus âgées ont tendance à avoir des dépenses médicales plus élevées.

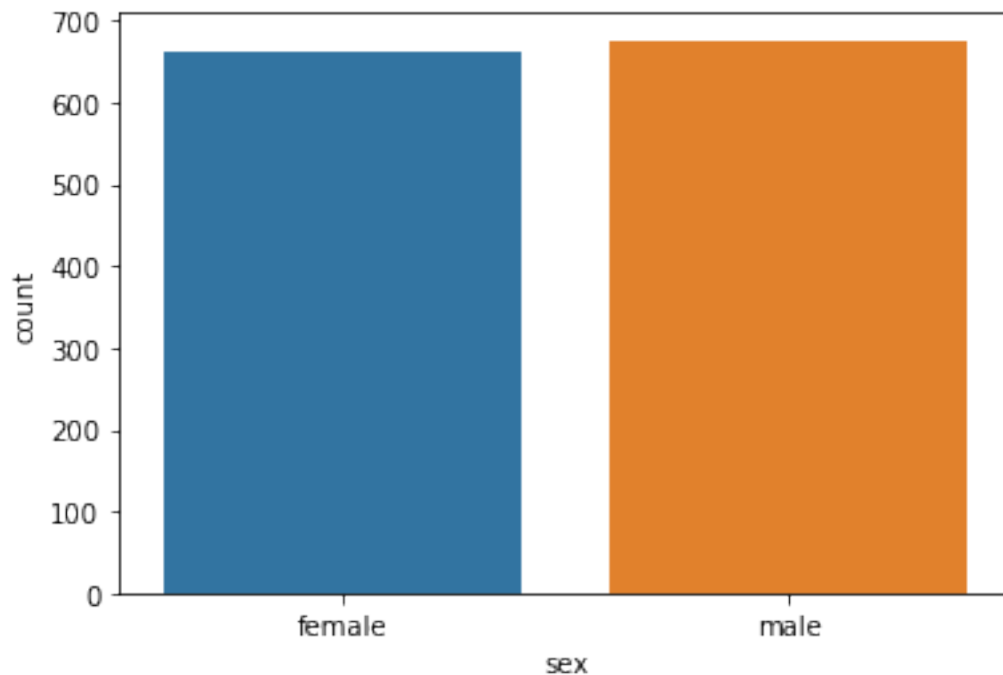
2-L'indice de masse corporelle (BMI) présente une corrélation faible avec l'âge, une corrélation très faible avec le nombre d'enfants (children), et une corrélation modérée avec les dépenses médicales (expenses). Cela suggère que des niveaux élevés d'IMC ne sont que modérément associés à des dépenses médicales plus élevées.

3-Le nombre d'enfants (children) présente une corrélation faible avec l'âge, une corrélation très faible avec l'indice de masse corporelle (bmi), et une corrélation faible avec les dépenses médicales (expenses). Cela suggère que le nombre d'enfants a un impact limité sur les dépenses médicales.

4-Les dépenses médicales (expenses) présentent une corrélation modérée avec l'âge, une corrélation modérée avec l'indice de masse corporelle (bmi), et une corrélation faible avec le nombre d'enfants (children). Cela suggère que l'âge et l'indice de masse corporelle sont des facteurs plus importants dans la détermination des dépenses médicales que le nombre d'enfants.

```
[12]: sns.countplot(x='sex', data=df)
```

```
[12]: <AxesSubplot:xlabel='sex', ylabel='count'>
```



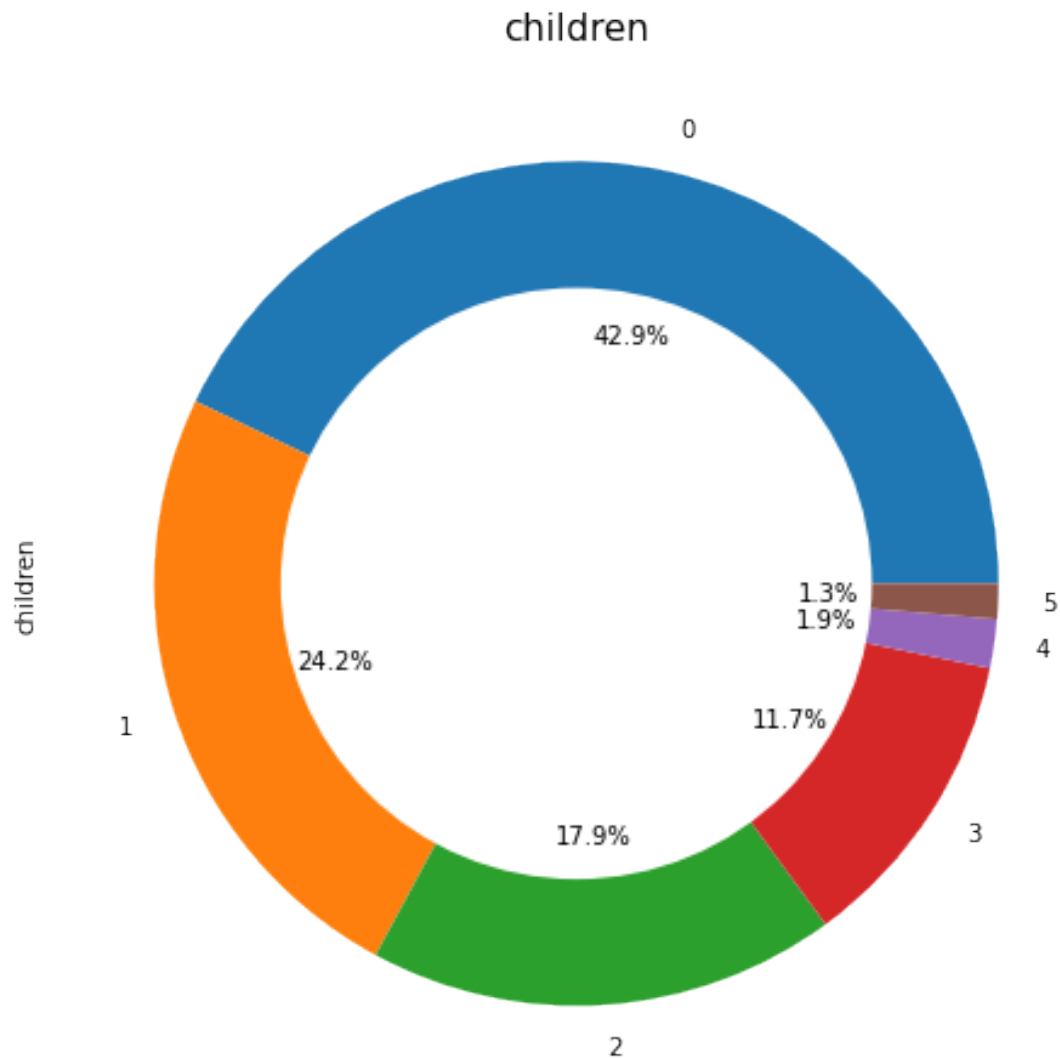
```
[13]: df.children.value_counts(normalize=True)*100
```

```
[13]: 0    42.899851
      1    24.215247
      2    17.937220
      3    11.733931
      4     1.868460
      5     1.345291
      Name: children, dtype: float64
```

```
[14]: plt.figure(figsize=(8,8))

df.children.value_counts().plot(kind='pie',autopct='%1.1f%%')
plt.title('children',fontsize='15')
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
```

```
[14]: <matplotlib.patches.Circle at 0x7ff6a89ad7c0>
```



```
[15]: df.region.value_counts(normalize=True)*100
```

```
[15]: southeast    27.204783
      southwest    24.289985
      northwest    24.289985
      northeast    24.215247
      Name: region, dtype: float64
```

```
[16]: plt.figure(figsize=(8,8))

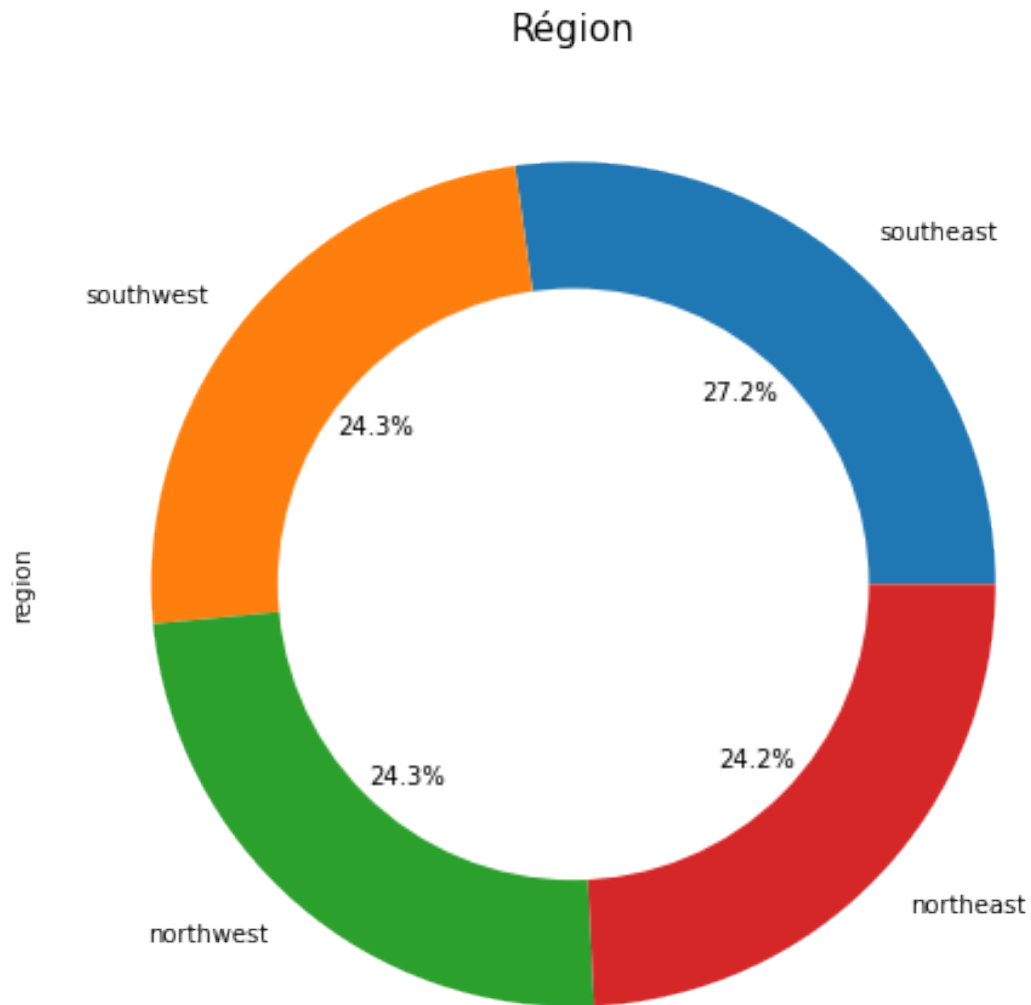
      df.region.value_counts().plot(kind='pie',autopct='%1.1f%%')
      plt.title('R gion',fontsize='15')
```

```

centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

```

[16]: <matplotlib.patches.Circle at 0x7ff6c8e009d0>



```

[17]: plt.figure(figsize=(8,8))

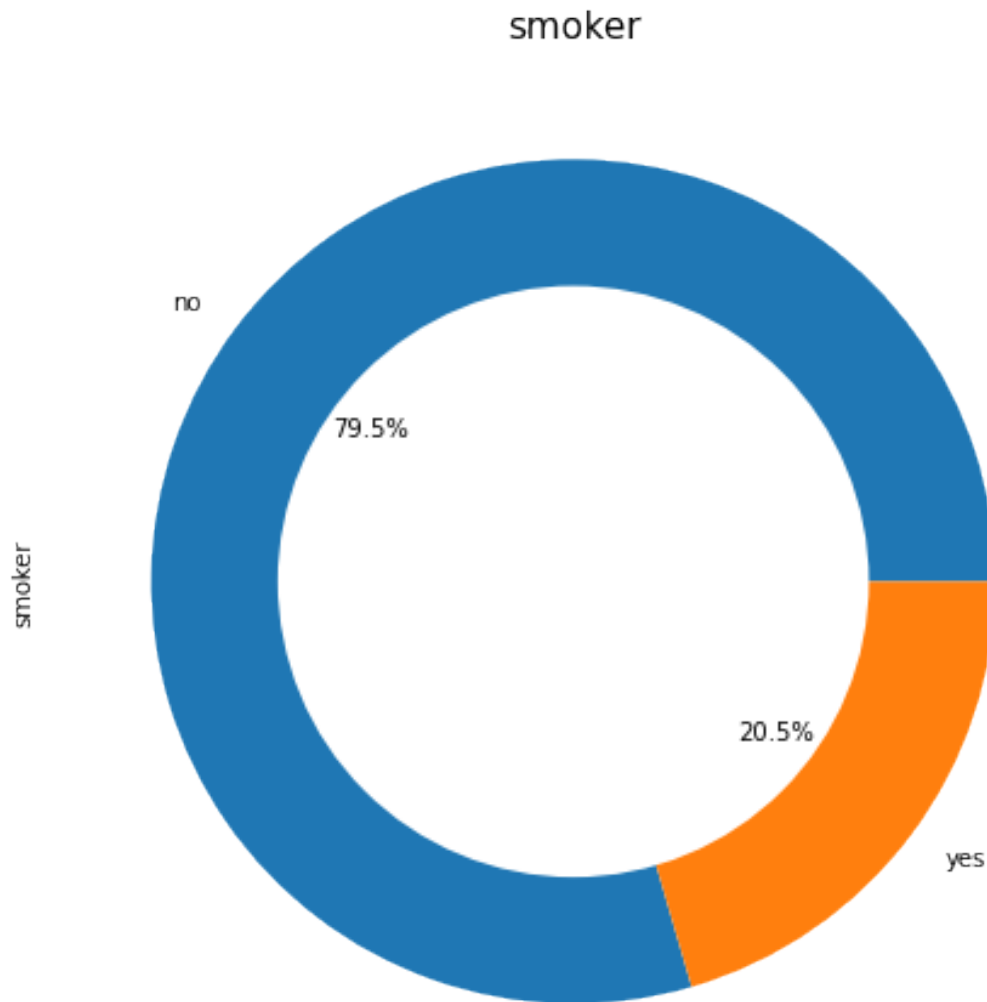
df.smoker.value_counts().plot(kind='pie',autopct='%1.1f%%')
plt.title('smoker',fontsize='15')
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()

```



```
fig.gca().add_artist(centre_circle)
```

```
[17]: <matplotlib.patches.Circle at 0x7ff6a8c85310>
```



```
[18]: df.groupby("smoker").expenses.agg(["mean", "median", "count"])
```

```
[18]:
```

	mean	median	count
smoker			
no	8434.268449	7345.405	1064
yes	32050.231971	34456.350	274

```
[19]: #df['smoker'].value_counts(normalize=True)*100  
df['smoker'].value_counts()/df.shape[0]*100
```

```
[19]: no      79.521674
      yes     20.478326
      Name: smoker, dtype: float64
```

```
[20]: df.duplicated().sum()
```

```
[20]: 1
```

```
[21]: df = df.drop_duplicates()
```

```
[22]: df.duplicated().sum()
```

```
[22]: 0
```

Détection des valeurs aberrantes :

Pour élimier les valeurs aberrantes, nous utilisons la technique selon laquelle :

$$\text{Valeur} < Q1 - 1.5 * \text{IQR} \text{ ou } \text{Valeur} > Q3 + 1.5 * \text{IQR}$$

$$\text{IQR} = Q3 - Q1$$

```
[23]: ## définir une fonction de détection des valeurs aberrantes

def outlier_detect(df, variable_name):
    IQR = df[variable_name].quantile(0.75) - df[variable_name].quantile(0.25)
    lower = df[variable_name].quantile(0.25) - 1.5*IQR
    upper = df[variable_name].quantile(0.75) + 1.5*IQR
    return df[(df[variable_name]<lower) | (df[variable_name]>upper)]
```

```
[24]: outlier_detect(df, 'expenses').sort_values('expenses')
```

```
[24]:
```

	age	sex	bmi	children	smoker	region	expenses
623	18	male	33.5	0	yes	northeast	34617.84
1078	28	male	31.7	0	yes	southeast	34672.15
223	19	male	34.8	0	yes	southwest	34779.62
689	27	male	31.1	1	yes	southeast	34806.47
1291	19	male	34.9	0	yes	southwest	34828.65
...	...	...	...	...	...	...	...
819	33	female	35.5	0	yes	northwest	55135.40
577	31	female	38.1	1	yes	northeast	58571.07
1230	52	male	34.5	3	yes	northwest	60021.40
1300	45	male	30.4	0	yes	southeast	62592.87
543	54	female	47.4	0	yes	southeast	63770.43

[139 rows x 7 columns]

```
[25]: outlier_detect(df, 'bmi').sort_values('bmi')
```

```
[25]:
```

	age	sex	bmi	children	smoker	region	expenses
543	54	female	47.4	0	yes	southeast	63770.43
401	47	male	47.5	1	no	southeast	8083.92
860	37	female	47.6	2	yes	southwest	46113.51
1088	52	male	47.7	1	no	southeast	9748.91
286	46	female	48.1	2	no	northeast	9432.93
116	58	male	49.1	0	no	southeast	11381.33
847	23	male	50.4	1	no	southeast	2438.06
1047	22	male	52.6	1	yes	southeast	44501.40
1317	18	male	53.1	0	no	southeast	1163.46

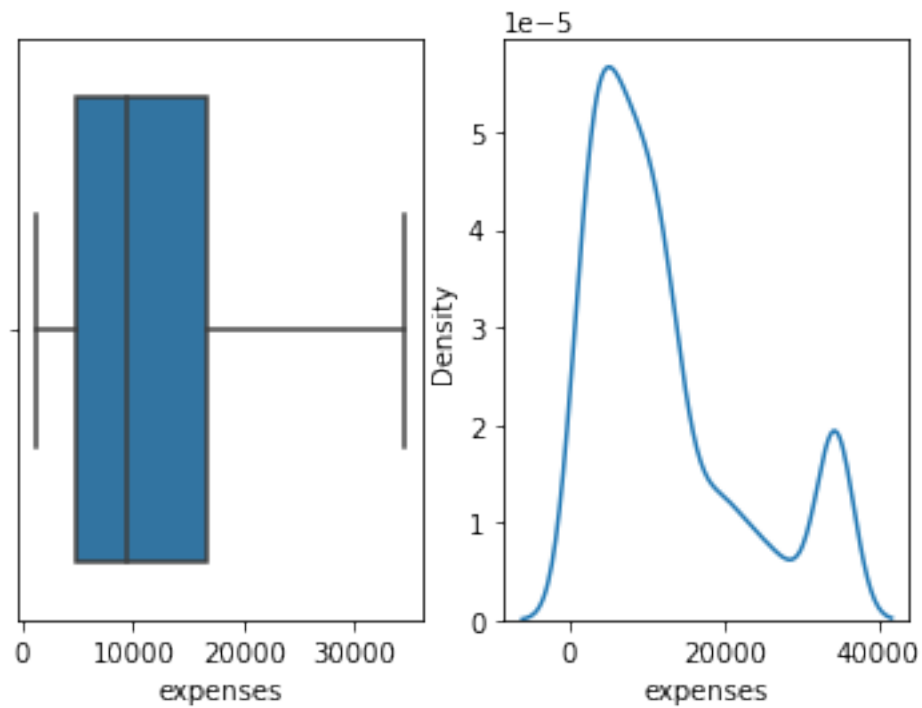
Remplaçons les valeurs aberrantes de la borne inférieure par  $Q1-1.5IQR$  et celles de la borne supérieure par  $Q3+1.5IQR$

```
[26]: df['expenses'].quantile(0.75)+1.5*(df['expenses'].quantile(0.75)-df['expenses'].
      ↪ quantile(0.25))
```

```
[26]: 34524.79
```

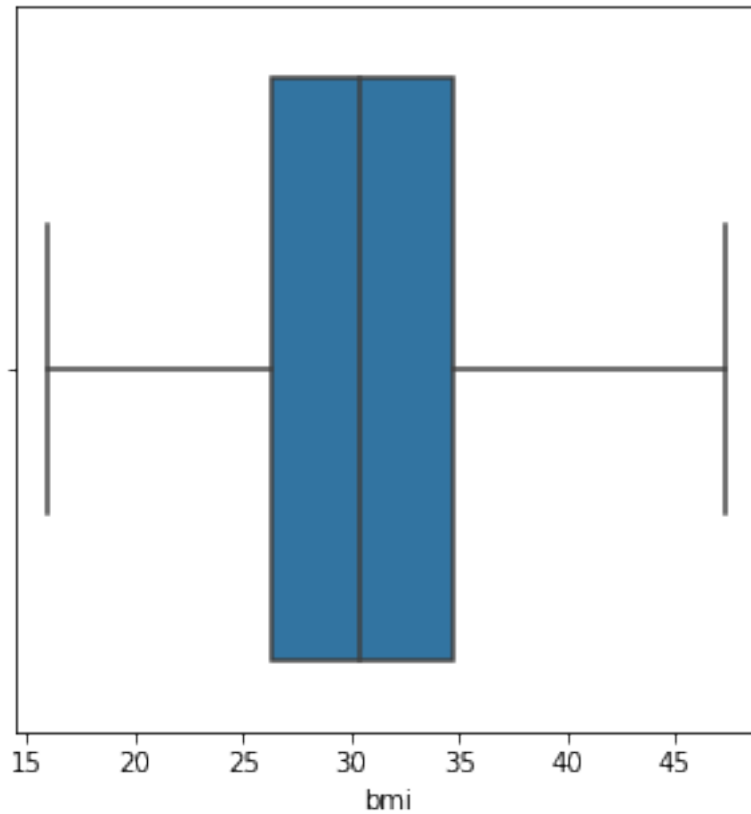
```
[27]: df.loc[(outlier_detect(df,'expenses').index, 'expenses')] =df['expenses'].
      ↪ quantile(0.75)+1.5*(df['expenses'].quantile(0.75)-
      ↪
      ↪ df['expenses'].quantile(0.25))
```

```
[28]: f, axes = plt.subplots(1, 2)
      sns.boxplot(df['expenses'], ax=axes[0])
      sns.kdeplot(df['expenses'], ax=axes[1])
      plt.show()
```



```
[29]: df.loc[(outlier_detect(df,'bmi').index, 'bmi')] =df['bmi'].quantile(0.75)+1.  
      ↪5*(df['bmi'].quantile(0.75)-  
      ↪df['bmi'].quantile(0.25))
```

```
[30]: plt.figure(figsize=(5,5))  
      sns.boxplot(df['bmi'])  
      plt.show()
```



### 0.3 Préparations des données

#### Séparer les variables catégorielles et numériques

[31]: *#Séparer les variables catégorielles et numériques*

```
cat_df = []
num_df = []
for j, i in enumerate(df.dtypes):
    if i == object:
        cat_df.append(df.iloc[:,j])
    else :
        num_df.append(df.iloc[:,j])
```

[32]: `cat_df = pd.DataFrame(cat_df).transpose()`  
`num_df = pd.DataFrame(num_df).transpose()`

[33]: `cat_df.head()`

```
[33]:      sex smoker  region
0  female    yes southwest
1   male    no  southeast
```

```

2    male    no    southeast
3    male    no    northwest
4    male    no    northwest

```

```
[34]: num_df.head()
```

```

[34]:   age    bmi  children  expenses
0  19.0  27.9      0.0  16884.92
1  18.0  33.8      1.0   1725.55
2  28.0  33.0      3.0   4449.46
3  33.0  22.7      0.0  21984.47
4  32.0  28.9      0.0   3866.86

```

```

[35]: num_df['children'] = num_df['children'].astype(int)
      num_df['age'] = num_df['age'].astype(int)

```

### transformations des variables categorielles en variables dichotomiques

```

[36]: #transformations des variables categorielles en variables dichotomiques
      labelencoder = LabelEncoder()

      for i in cat_df:
          cat_df[i] = labelencoder.fit_transform(cat_df[i])
      cat_df

```

```

[36]:   sex  smoker  region
0      0       1       3
1      1       0       2
2      1       0       2
3      1       0       1
4      1       0       1
...    ...    ...    ...
1333   1       0       1
1334   0       0       0
1335   0       0       2
1336   0       0       3
1337   0       1       1

```

[1337 rows x 3 columns]

```
[37]: cat_df['sex'].nunique()
```

```
[37]: 2
```

```
[38]: cat_df['smoker'].nunique()
```

```
[38]: 2
```

```
[39]: cat_df['region'].nunique()
```

```
[39]: 4
```

```
[40]: data = pd.concat([num_df, cat_df], axis =1)
```

```
[41]: data.head()
```

```
[41]:
```

	age	bmi	children	expenses	sex	smoker	region
0	19	27.9	0	16884.92	0	1	3
1	18	33.8	1	1725.55	1	0	2
2	28	33.0	3	4449.46	1	0	2
3	33	22.7	0	21984.47	1	0	1
4	32	28.9	0	3866.86	1	0	1

```
[42]: X = data.drop('expenses', axis=1)
      y = data['expenses']

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
      ↪random_state = 123)

      X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size = 0.
      ↪2, random_state = 123)
```

```
[43]: X.shape
```

```
[43]: (1337, 6)
```

```
[44]: y.shape
```

```
[44]: (1337,)
```

## 0.4 Modélisations

### 0.4.1 Modèle de régression linéaire

```
[45]: model_reg = LinearRegression()

      model_reg.fit(X_train, y_train)
      model_reg.score(X_train, y_train)
      #model.predict(X_train)

      print(model_reg.intercept_)

      print(model_reg.coef_)

      print(model_reg.score(X_test, y_test))
```

```
-6949.413155648976  
[ 233.19251789  210.29765071  462.41384768 -164.0645531  
 19723.08258238 -355.18699393]  
0.8133343561896759
```

#### 0.4.2 Modèle de Ridge Regression

```
[46]: model_ridge = linear_model.Ridge(alpha=.5)
```

```
model_ridge.fit(X_train, y_train)  
model_ridge.score(X_train, y_train)  
#model.predict(X_train)
```

```
print(model_ridge.intercept_)
```

```
print(model_ridge.coef_)
```

```
print(model_ridge.score(X_test, y_test))
```

```
-6938.418261357763  
[ 233.15278515  210.31589157  461.85220107 -159.58724772  
 19666.42466037 -355.13765316]  
0.8134101609303416
```

#### 0.4.3 Modèle de Lasso Regression

```
[47]: model_lasso = linear_model.Ridge(alpha=.1)
```

```
model_lasso.fit(X_train, y_train)  
model_lasso.score(X_train, y_train)  
#model.predict(X_train)
```

```
print(model_lasso.intercept_)
```

```
print(model_lasso.coef_)
```

```
print(model_lasso.score(X_test, y_test))
```

```
-6947.209538455327  
[ 233.18455343  210.30130451  462.30128025 -163.1658135  
 19711.72476242 -355.17717729]  
0.813350459813973
```

```
[48]: ##Prédire les dépenses sur le meilleur modèle
```

```
y_test_pred = model_ridge.predict(X_test)
```

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_test_pred})  
df
```



[48] :	Actual	Predicted
31	2198.19	2789.639820
188	6775.96	8789.456880
631	1977.82	3594.503998
328	34524.79	34154.901026
55	34524.79	34441.535427
681	1242.26	535.897048
209	6610.11	11354.972831
1284	34524.79	33821.645151
523	5397.62	9140.021380
723	1263.25	3711.667011
1002	1972.95	2858.398378
314	34524.79	24561.637634
1015	12124.99	11665.926397
973	1759.34	4943.865306
1193	8671.19	11595.339404
309	7749.16	10150.868690
260	11837.16	10818.990786
332	13429.04	13490.619797
13	11090.72	13778.434885
306	20177.67	5231.838184
261	17085.27	22779.104891
832	4719.74	5163.944691
136	1261.44	3438.256352
1198	6393.60	8518.107657
210	1980.07	3901.913857
409	4074.45	6423.937258
1114	2396.10	3417.247893
1120	34166.27	23629.026493
1085	19023.26	26913.593882
1100	16776.30	25362.786240
601	9174.14	11083.631054
895	13063.88	15472.387903
1126	10214.64	10948.429873
502	22218.11	29090.116773
45	20630.28	12504.767471
1041	1704.70	1957.041719
550	13390.56	13002.936457
66	14235.07	15365.544436
1171	22478.60	28227.301908
1058	2480.98	6254.450993
1212	1702.46	1620.536292
707	10264.44	11392.966002
226	2689.50	6733.032638
595	8823.99	11336.107603
1150	2203.74	3630.903386
351	8932.08	9037.894861

1065	7045.50	7571.430013
1299	2710.83	3003.317618
874	8891.14	9152.191666
396	8522.00	10642.161814
830	13393.76	13486.663007
1043	3161.45	3950.596766
169	4827.90	4557.646644
145	5138.26	8658.550398

[ ]: