UNIVERSITÉ PARIS 8
VINCENNES-SAINT-DENIS

Prédiction de consommation d'électricité en France

Par un **modèle de régression linéaire**

Master 1 Mathématiques et applications
(Cybersécurité et sciences des données)

BENYOUCEF Asma

DIALLO Mouctar

TISSEMLAL Sara

1 Introduction

2 Nettoyage, Analyse et préduction

3 Collecte et pré-traitement de données de consommation

4 Collecte et pré-traitement de données de production

5 Création du dataset dju chauffage mensuelle en France

6 Un peut de statistiques sur nos données

7 Création d'un échantillon de travail par jointure des deux précédents dataframe

8 Régression Linéaire avec SKLearn

9 Régression Linéaire avec Statsmodels

10 Série temporelle avec le modèle additif

11 Prévision par lissage exponentiel par la méthode de Holt-Winters

12 Conclusion

Introduction

En France, l'électricité représentait plus de 24% de la consommation énergétique en 2019.

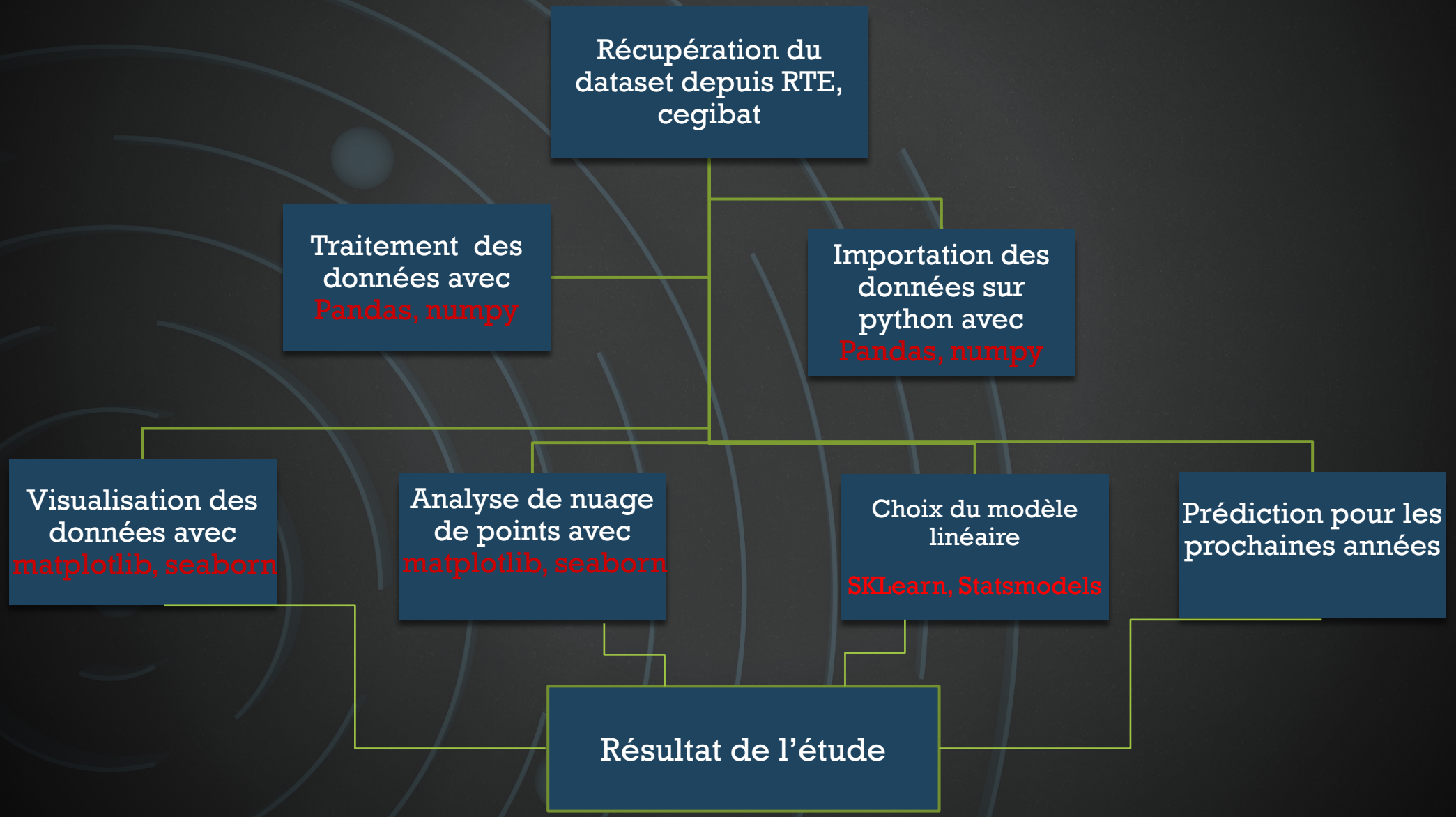
Cette consommation est en partie influencée par la température avec l'utilisation des chauffages électriques par les ménages et les entreprises.

Comme l'électricité n'est pas stockable, la production de cette dernière suit la loi de l'offre et de la demande française mais aussi européenne.

Problématique

L'année 2022 est marquée par une crise énergétique sans précédent, la France risque une pénurie d'électricité pour les prochaines années, c'est pourquoi nous allons analyser la consommation et la production d'électricité des 10 dernières années en fonction des DJU du chauffage afin de prédire ce qui se passerait pour les prochaines années.

ÉTAPES DE LA RÉALISATION DU PROJET



Les librairies utilisées

```
# -*- coding: utf-8 -*-
#importer les librairies dont nous aurons besoin pour ce projet
#Matplotlib est la librairie qui permet de visualiser nos Datasets, nos fonctions, nos résultats sous forme de gr
import matplotlib.pyplot as plt
#pandas pour l'extraction des données et le traitement
import pandas as pd
#Numpy pour manipuler notre Dataset en tant que matrice
import numpy as np
import statsmodels.api as sm

#our l'affichage des donnée tableau
from IPython.display import display

#Calculez les quantiles d'un diagramme de probabilité et affichez éventuellement le diagramme
from statsmodels.graphics.gofplots import ProbPlot

#pour ignorer tous les avertissements qui pourraient être générés pendant l'exécution du code
import warnings
warnings.filterwarnings('ignore')

from matplotlib.pyplot import figure
```

APERÇU DES DONNÉES DE: www.rte-france.com

Entrée [2]:

```
#Traitement du jeu de donnée

#Chargement du dataset 'new_data - new_data.csv'
data_coso_prod=pd.read_csv("new_data - new_data.csv")
#afficher les 10 prémière ligne des données importées
data_coso_prod.iloc[35:].head(15)
```

	Mois	Qualité	Territoire	Production totale	Production nucléaire	Production thermique totale	Production thermique charbon	Production thermique fioul	Production thermique gaz	Production hydraulique	...	Consommation totale	Solde exportateur	Echanges export	Echange impo
35	2012-12	Données définitives	France	54715	39912.0	5319.0	1714.0	548.0	3057.0	6521.0	...	49602	4367.0	6716.0	2349.
36	2013-01	Données définitives	Grand-Est	9924	7258.0	1508.0	928.0	8.0	570.0	814.0	...	0	0.0	NaN	Nal
37	2013-01	Données définitives	Nouvelle-Aquitaine	5644	4801.0	187.0	NaN	0.0	186.0	465.0	...	0	0.0	NaN	Nal
38	2013-01	Données définitives	Auvergne-Rhône-Alpes	13119	9641.0	278.0	0.0	49.0	227.0	3047.0	...	0	0.0	NaN	Nal
39	2013-01	Données définitives	Bourgogne-Franche-Comté	403	NaN	244.0	75.0	0.0	168.0	121.0	...	0	0.0	NaN	Nal
40	2013-	Données	Bretagne	246	NaN	49.0	NaN	1.0	48.0	56.0		0	0.0	NaN	Nal

Source:
https://www.rte-france.com/eco2mix/telecharger-les-indicateurs

Traitement du jeu de donnée de la consommation

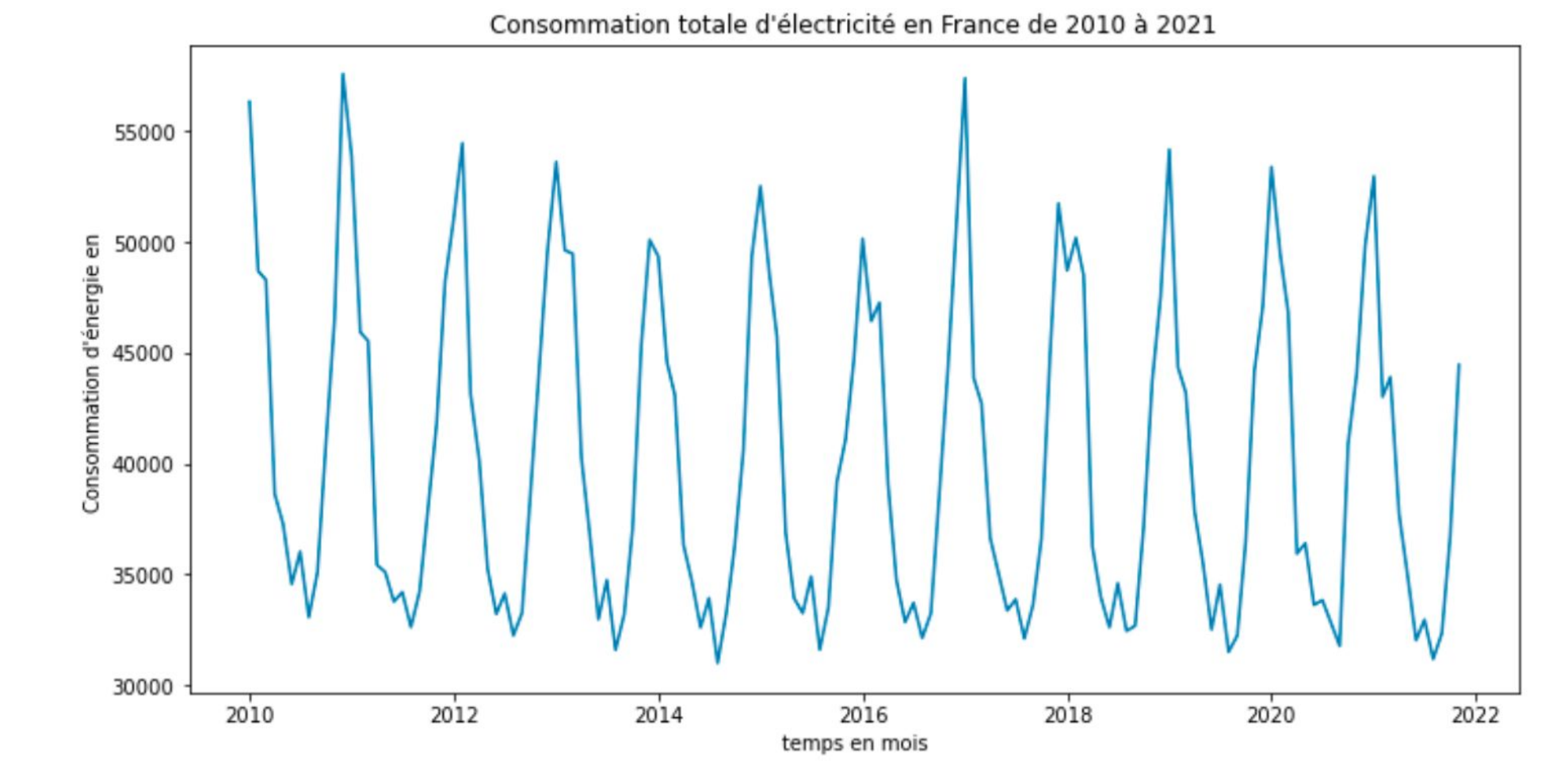
Entrée [3]:

```
#On prend juste les données de la france
#Copier dans consommation
consommation=data_coso_prod.copy()
#Trier et extraire seulement ce qui nous intéresse
consommation=consommation[['Mois', 'Territoire', 'Consommation totale']]
#On récupère les information où Territoire=France et on laisse les données des régions
consommation=consommation[consommation['Territoire']=='France']
#Eliminer les données manquantes
consommation.drop('Territoire', axis=1, inplace=True)
#renommer les colonnes Mois et Consommation Totale
consommation.rename(columns={'Mois': 'mois', 'Consommation totale': 'consommationT'}, inplace=True)
#Aperçu des données mensuelles de consommation totale d'électricité de notre copy consommation
display(consommation.head())
#nobre de ligne et de colonne
display(consommation.shape)
```

	mois	consommationT
0	2010-01	56342
1	2010-02	48698
2	2010-03	48294
3	2010-04	38637
4	2010-05	37284

```
(143, 2)
```

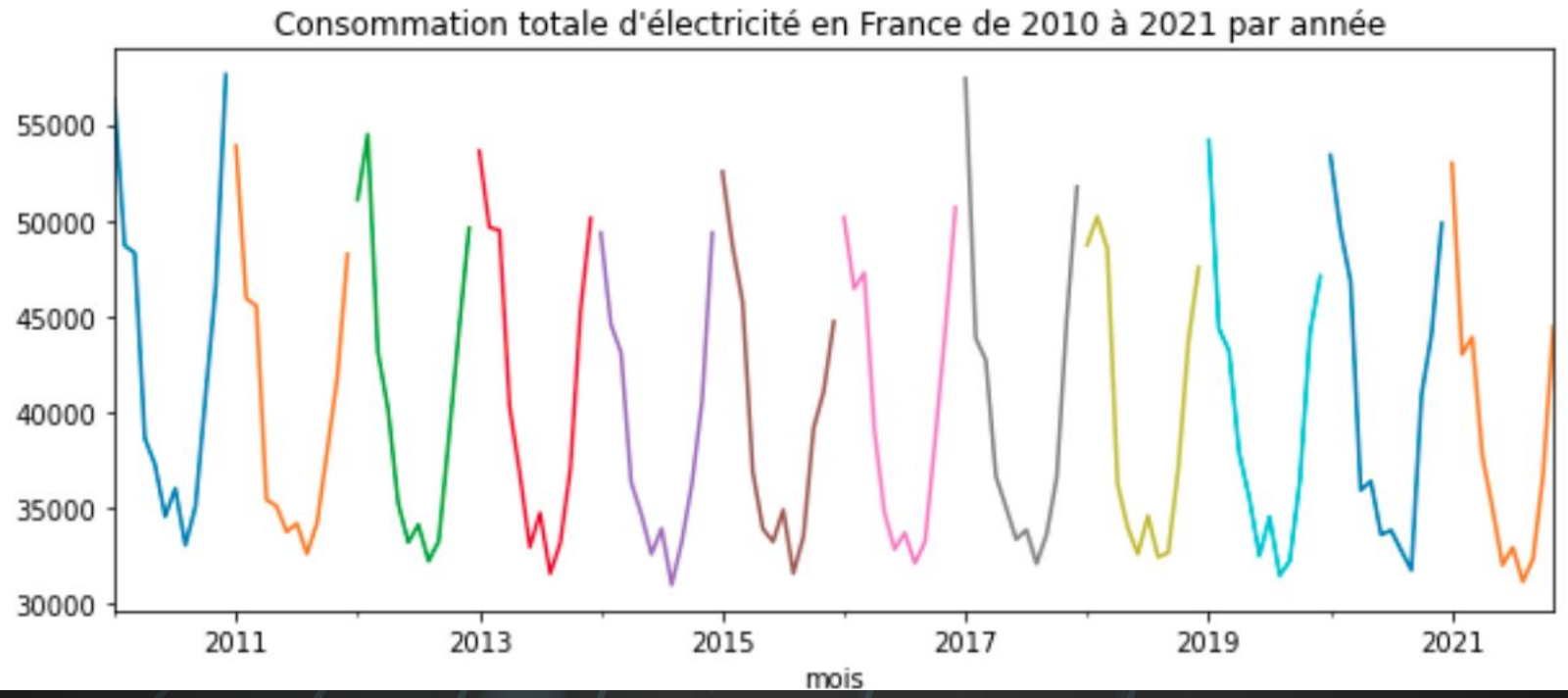
Visualisation de la consommation d'électricité en France



Visualisation de la consommation totale d'électricité en France

Entrée [64]:

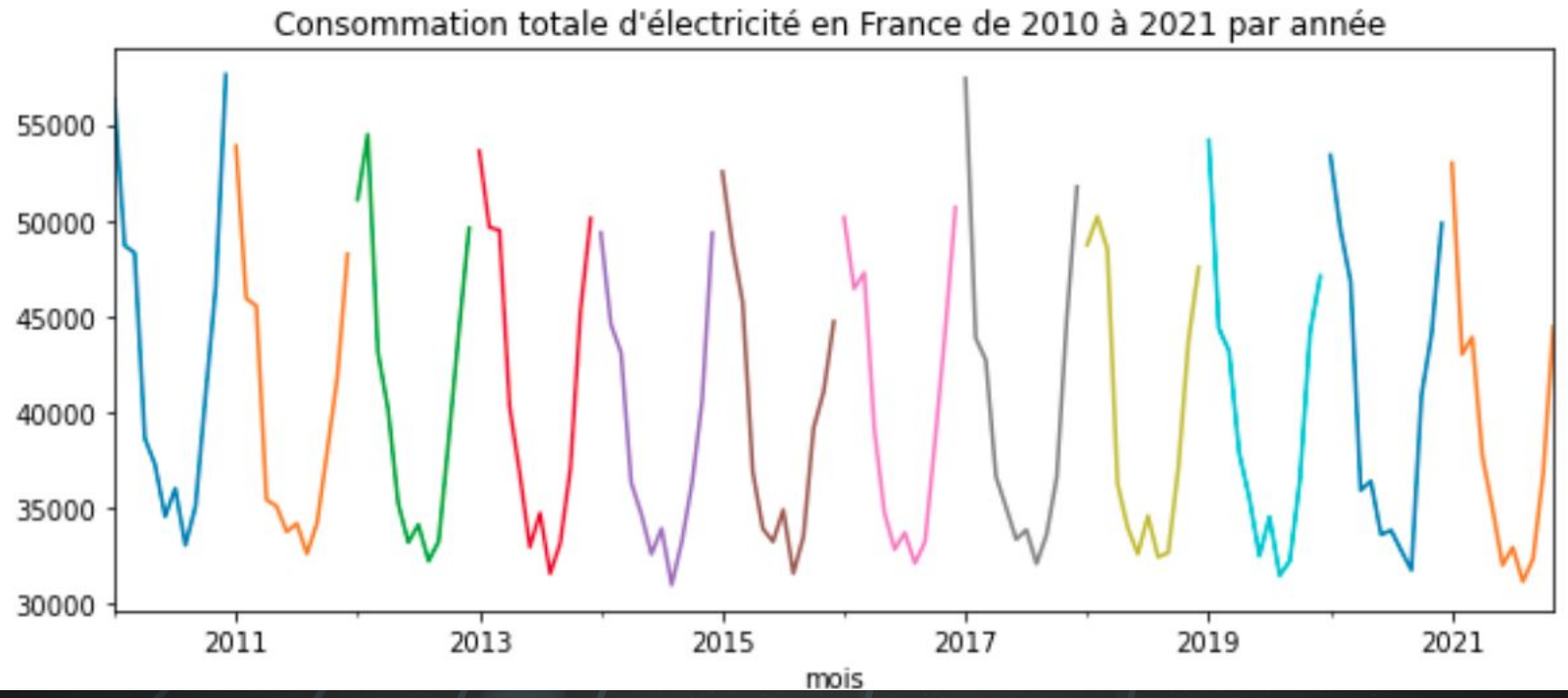
```
#Consommation totale d'électricité en France de 2010 à 2021 par année
figure(figsize=(10, 4))
consommation['consommationT'].resample('Y').plot(title='Consommation totale d\'électricité
plt.show()
```



Visualisation de la consommation totale d'électricité en France

Entrée [64]:

```
#Consommation totale d'électricité en France de 2010 à 2021 par année
figure(figsize=(10, 4))
consommation['consommationT'].resample('Y').plot(title='Consommation totale d\'électricité
plt.show()
```



Traitement du jeu de donnée de la production d'électricité

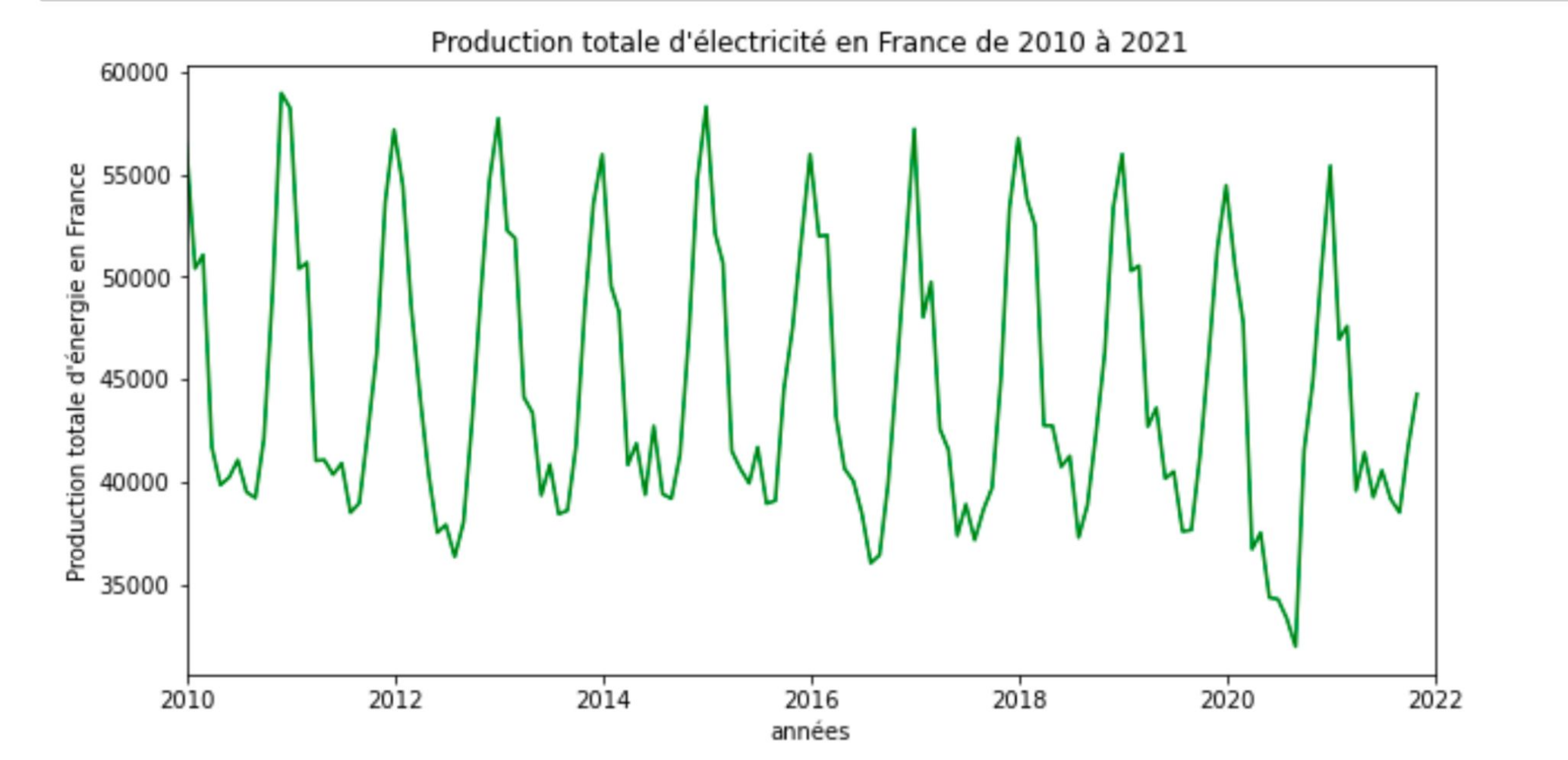
Entrée [10]:

```
#On prend juste les données de la production d'électricité la france
#Copier dans production
production=data_coso_prod.copy()
#Trier et extraire seulement ce qui nous intéresse
production=production[['Mois', 'Territoire', 'Production totale']]
#On récupère les information où Territoire=France et on laisse les données des régions
production=production[production['Territoire']=='France']
#Eliminer les données manquantes
production.drop('Territoire', axis=1, inplace=True)
#renommer les colonnes Mois et Production totale
production.rename(columns={'Mois': 'mois', 'Production totale': 'productionT'}, inplace=True)
#Aperçu des données mensuelles de consommation totale d'électricité de notre copy consommation
display(production.head())
#nobre de ligne et de colonne
display(production.shape)
```

	mois	productionT
0	2010-01	56542
1	2010-02	50406
2	2010-03	51071
3	2010-04	41693
4	2010-05	39847

```
(143, 2)
```

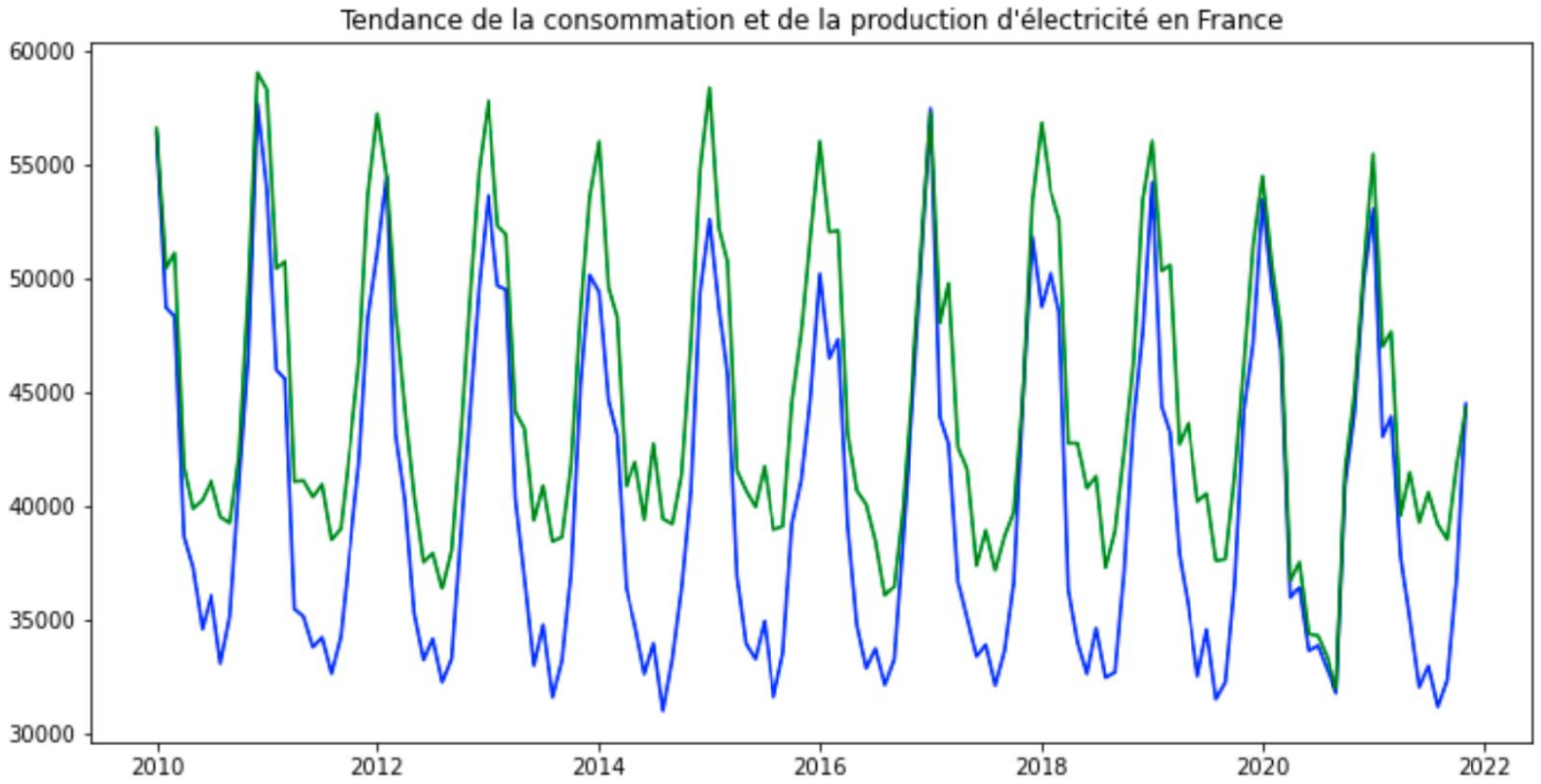
Visualisation de la production d'électricité en France



Tendance de la consommation et de la production d'électricité en France

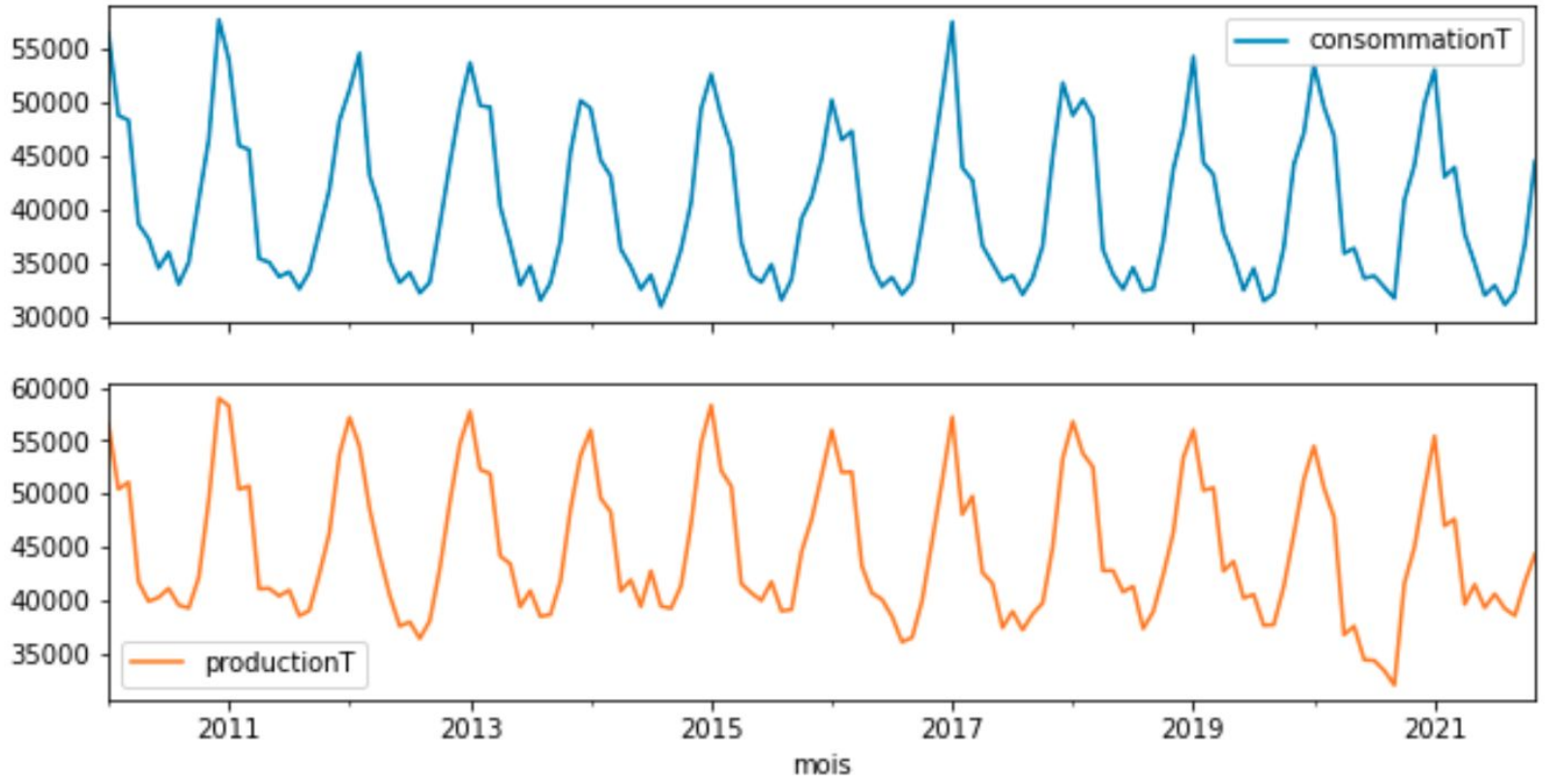
[66]:

```
figure(figsize=(12, 6))
plt.plot(consommation.consommationT, c='b')
plt.title('Tendance de la consommation et de la production d\'électricité en France')
plt.plot(production.productionT, c='g')
plt.show()#Tracer le graphe
```



Tendance de la consommation et de la production d'électricité en France

```
#Visualisation de la consommation et de la production
conso_prod[['consommationT', 'productionT']].plot(subplots=True, figsize=(10, 5))
plt.show()
```



Tendance de la consommation et de la production d'électricité en France

En général, il y a toujours un équilibre entre l'offre et la demande d'électricité en France, mais il peut y avoir des fluctuations au cours du temps en fonction de différents facteurs, tels que la météo, les besoins énergétiques saisonniers, l’importation et l'exportation... La France produit environ 500 GWh d'électricité par an, tandis que sa consommation est d'environ 550 GWh par an. La différence est comblée par des importations d'électricité.

C'est pour quoi, pour faire une prévision de la demande en électricité on va utiliser les données de météo france sur l'utilisation du chauffage en France.

Importation du dataset dju chauffage mensuelle en France

```
#Voyons l'impact du chauffage électrique
#Chargement du dataset 'dju.xlsx'
data_chauff = pd.read_csv('chauff_calcul_DJU.xlsx - DJU - Mensuel.csv', header=None, skiprows=11)
#A partir de la ligne 11 du file excel
data_chauff.head()
```

']:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	NaN	JAN	FÉV	MAR	AVR	MAI	JUN	JUI	AOÛ	SEP	OCT	NOV	DÉC	Total
1	2022.0	385.4	269.8	226.7	171.9	41	1.7	0	0	0	0	0	0	1096.4
2	2021.0	396.7	302.8	271	228.3	138.3	11.1	0.2	5.1	21	150.7	310.8	323.8	2159.4
3	2020.0	339	249.6	268.6	81.4	65.7	20.6	0.9	4.5	34.3	157.5	227.2	336.8	1785.9
4	2019.0	404.9	268.3	233.1	168.5	117.9	24.4	0	1.7	26.7	133.7	282.6	327.3	1989

Source: https://cegibat.grdf.fr/simulateur/calcul-dju

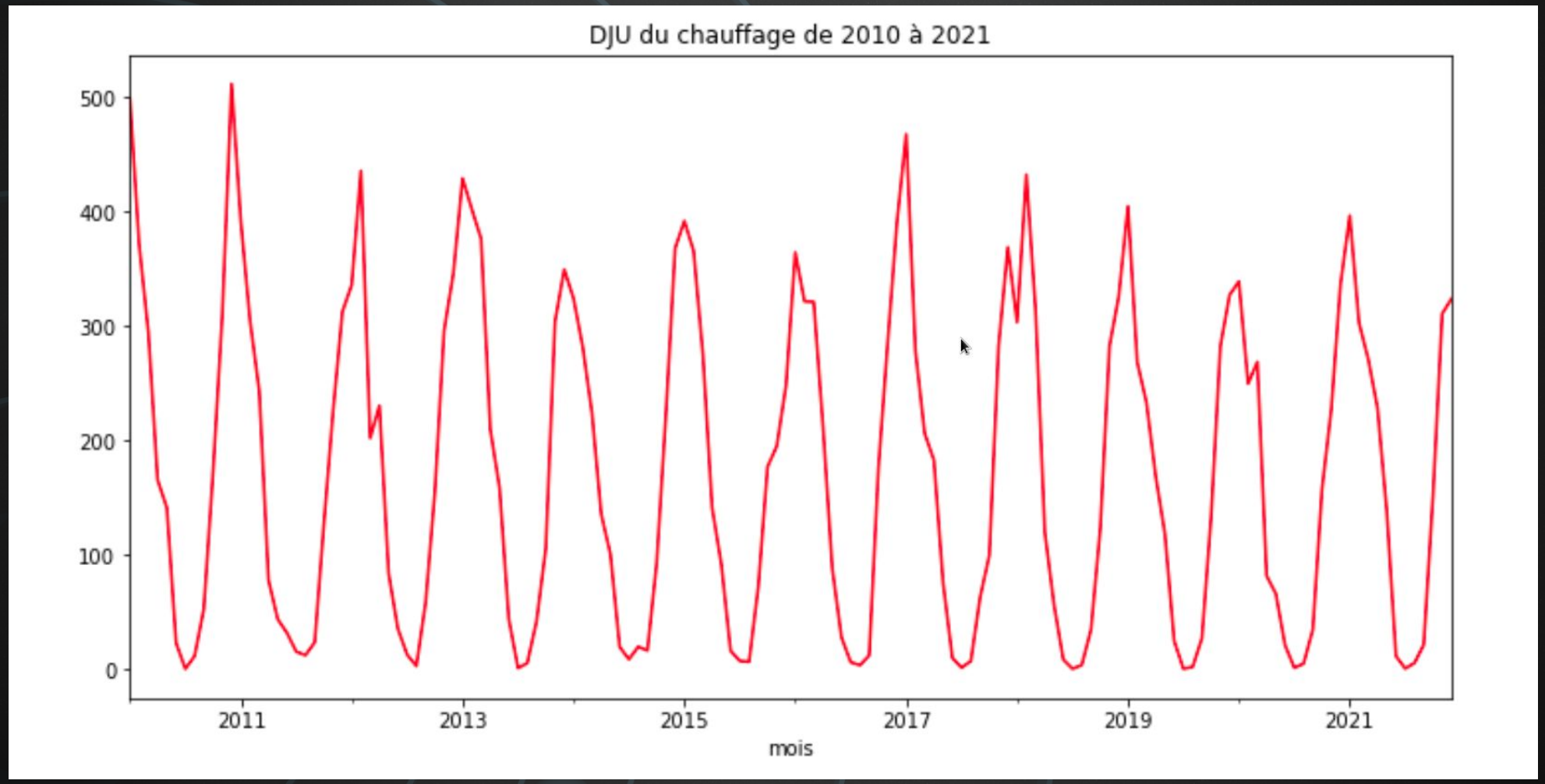
Traitement du dataset dju chauffage mensuelle en France

```python
#Il est nécessaire de transformer ces données de manière à obtenir une matrice temporelle.
#Transformation des données
dju = {'mois':[],'dju_chauffage':[]}
for year in data_chauff.index.values:
    for month in data_chauff.columns:
        dju['mois'].append(f"{year}-{month}")
        dju['dju_chauffage'].append(data_chauff.loc[year,month])
dju = pd.DataFrame(dju)
dju.head()
```

58]:

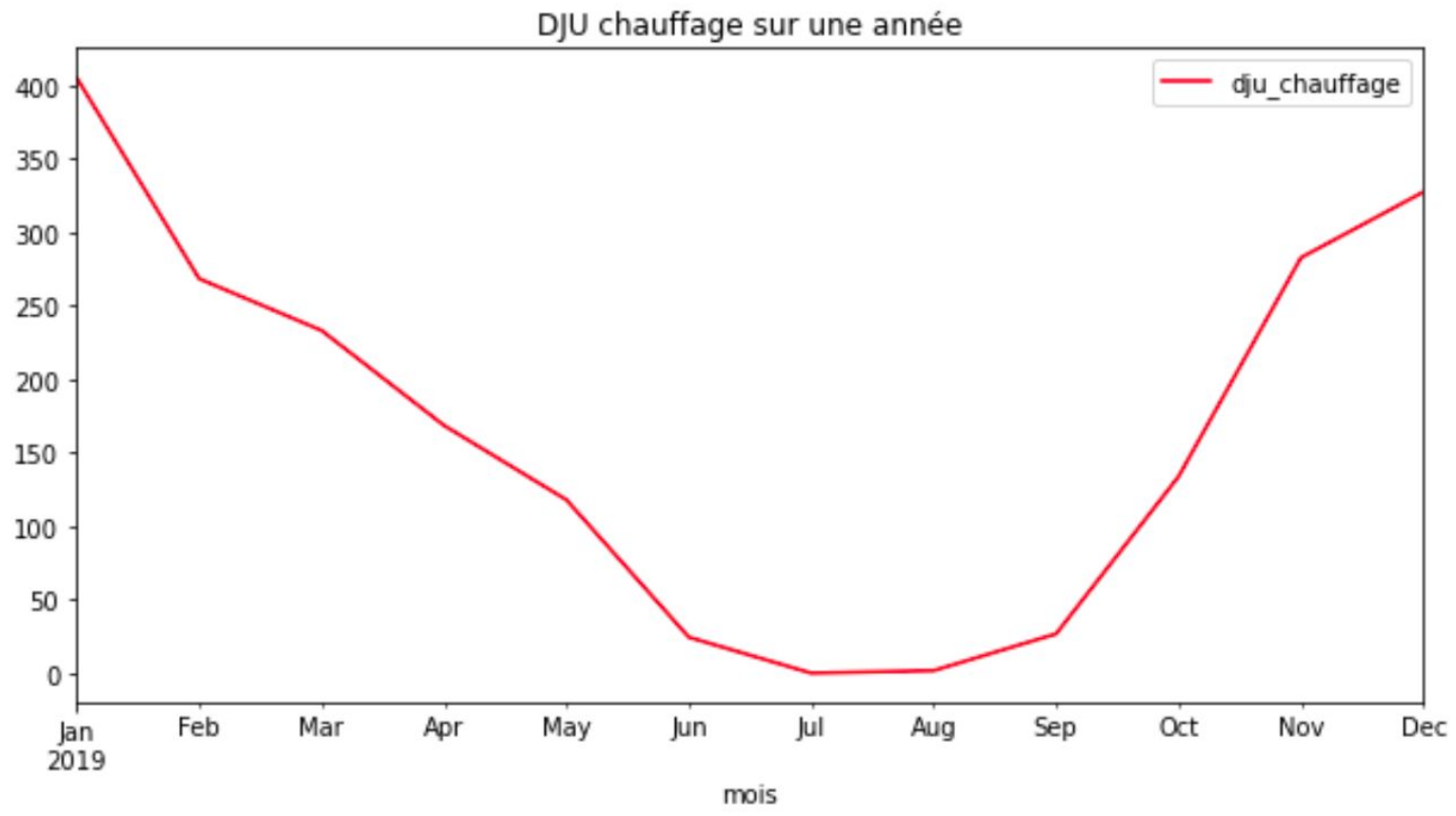
	mois	dju_chauffage
0	2021-1	396.7
1	2021-2	302.8
2	2021-3	271
3	2021-4	228.3
4	2021-5	138.3

Visualisation des données mensuelles des Degrés Jour Unifiés (DJU) chauffage électrique



Visualisation des données mensuelles des Degrés Jour Unifiés (DJU) chauffage électrique

```
#Courbe sur l'année 2019
dju_donnee['2019'].plot(figsize=(10, 5), title='DJU chauffage sur une année', color='r')
plt.show()
```



Statistiques des données

```
#Statistiques descriptives de l'échantillon dju
dju_donnee.describe().T
```

]:

	count	mean	std	min	25%	50%	75%	max
dju_chauffage	144.0	173.346528	143.640781	0.0	26.125	156.05	302.95	512.0

```
#Statistiques descriptives de l'échantillon de la consommation d'életricité en France
consommation.describe().T
```

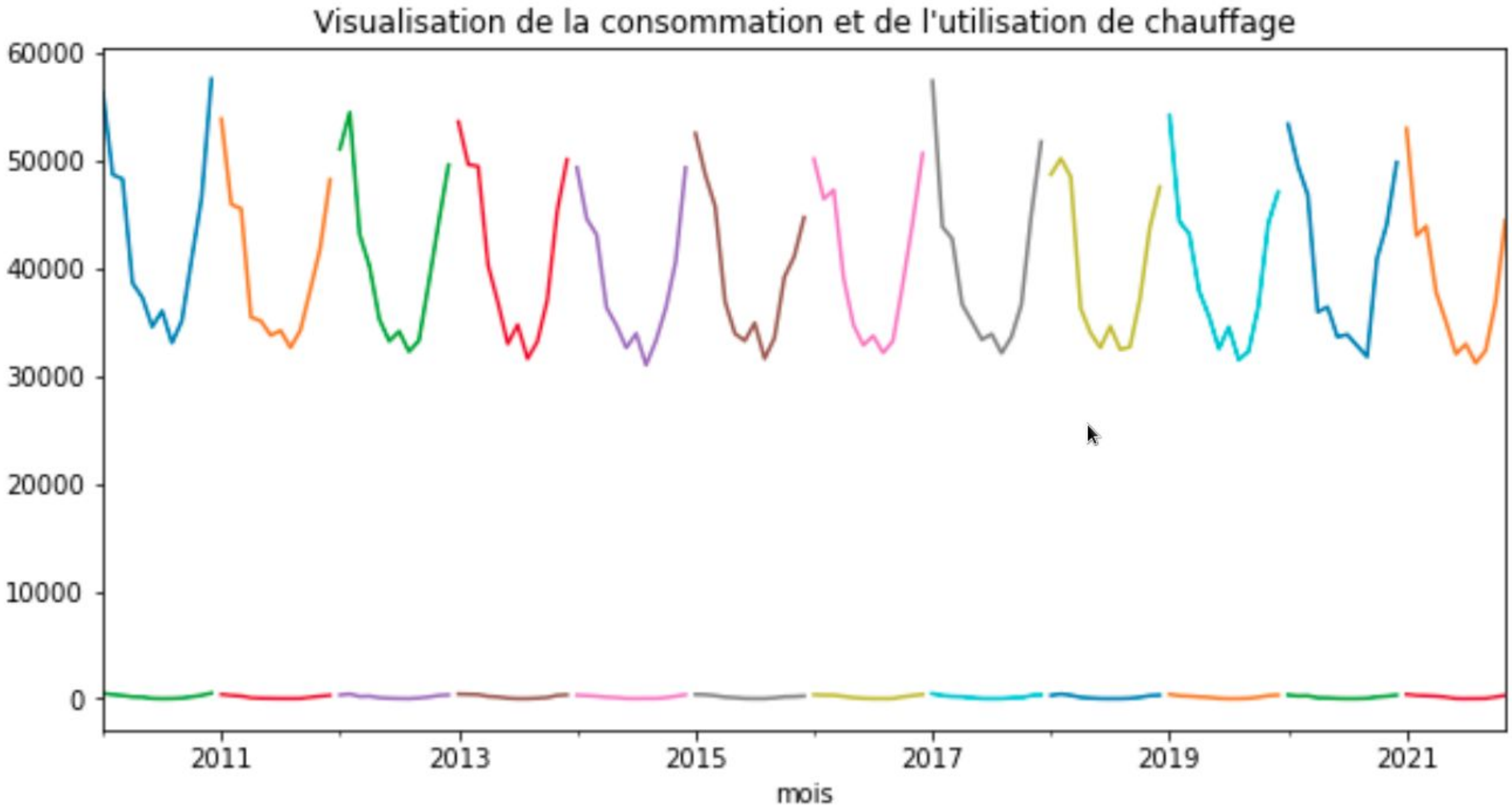
]:

	count	mean	std	min	25%	50%	75%	max
consommationT	143.0	40183.048951	7202.641671	31004.0	33858.5	37284.0	45838.0	57600.0

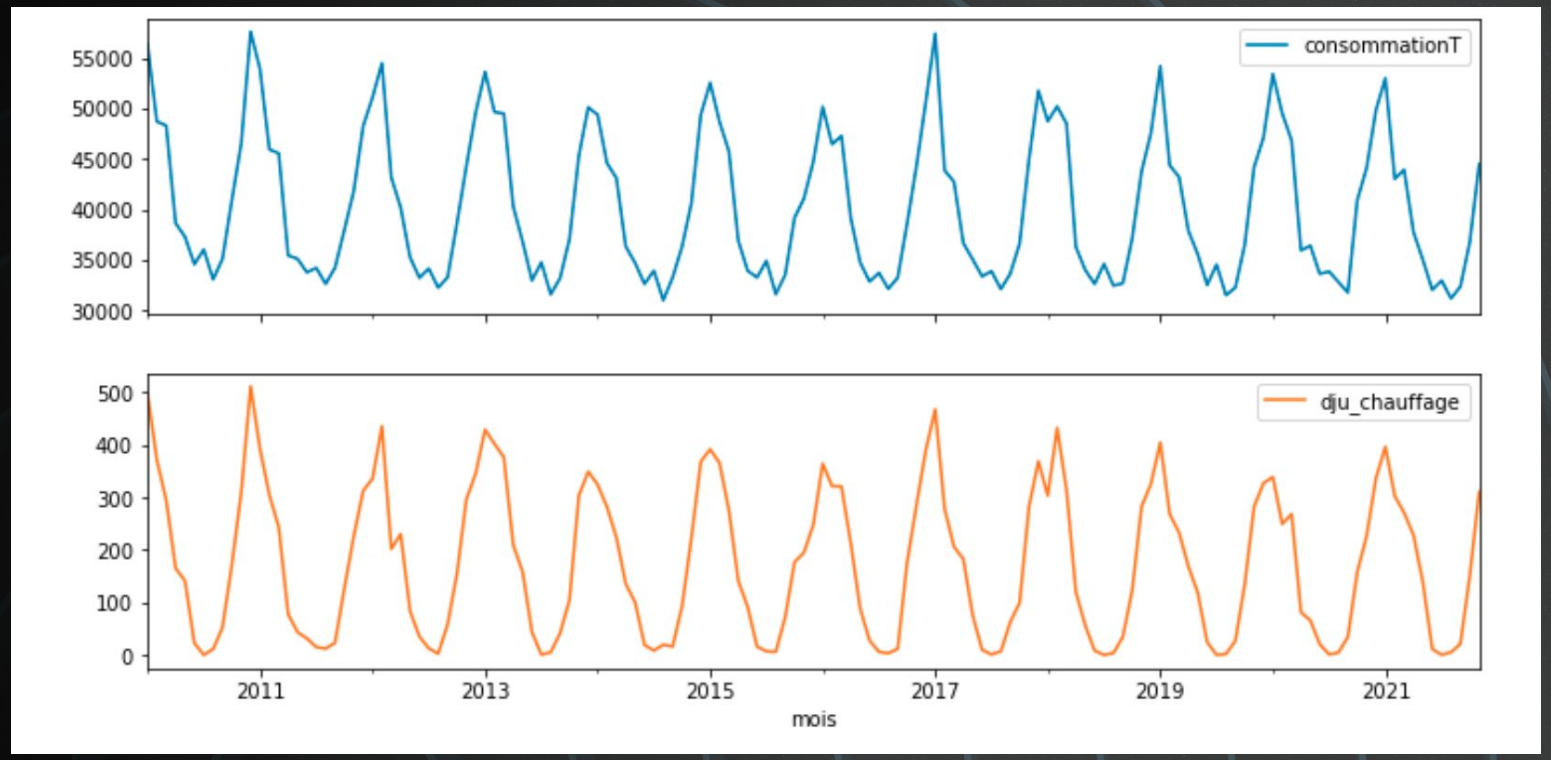
Création d'un échantillon de travail par jointure des deux précédents dataframe

```python
#inner = prendre les donnée de meme longueure comme clé "mois" avace la fonction merge
conso_dju = pd.merge(consommation, dju_donnee, how='inner', on='mois')
print(conso_dju)
```

```
            consommationT  dju_chauffage
mois
2010-01-01          56342          499.2
2010-02-01          48698          371.4
2010-03-01          48294          294.5
2010-04-01          38637          165.3
2010-05-01          37284          140.9
...                   ...            ...
2021-07-01          32959            0.2
2021-08-01          31184            5.1
2021-09-01          32350           21.0
2021-10-01          36762          150.7
2021-11-01          44458          310.8

[143 rows x 2 columns]
```


Visualisation de la consommation et de l'utilisation de chauffage
60000
50000
40000
30000
20000
10000
0
2011
2013
2015
2017
2019
2021
mois

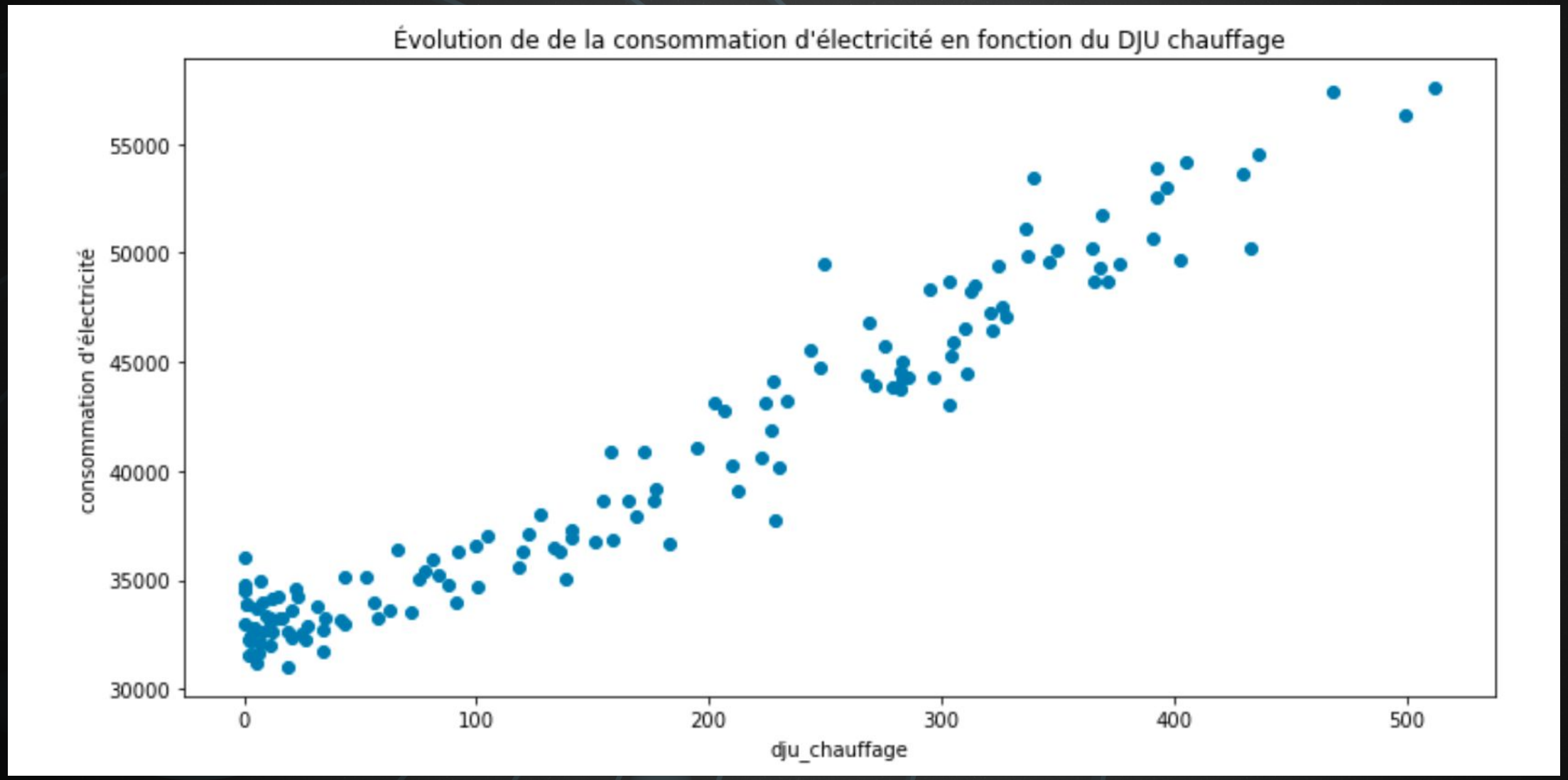
Comme la lecture est compliqué, on va les afficher séparément en respectant les échelles pour voir la tendance avec "subplots=True"




```
#La matrice de correlation
conso_dju[['consommationT', 'dju_chauffage']].corr()
```

	consommationT	dju_chauffage
consommationT	1.000000	0.970438
dju_chauffage	0.970438	1.000000

Nuage de point de l’évolution de de la consommation d'électricité en fonction du DJU chauffage



Régression Linéaire avec SKLearn

```python
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

```python
#conso_dju
```

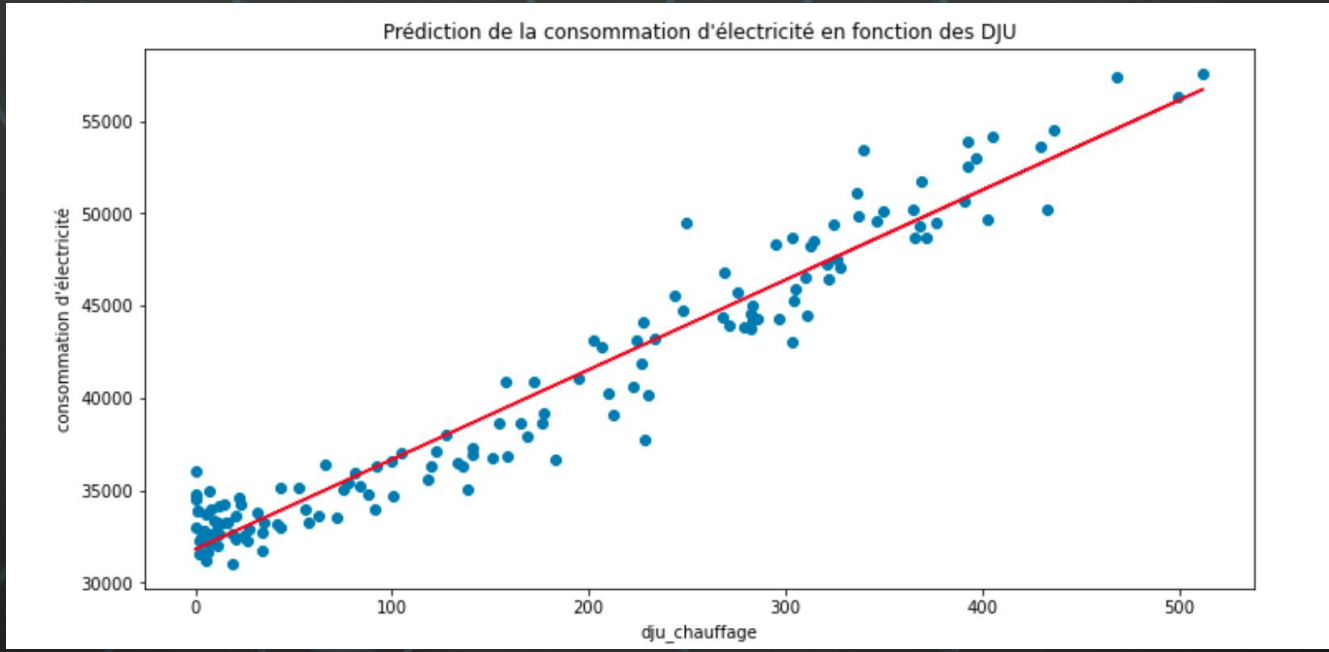
```python
y=conso_dju['consommationT']
X=conso_dju.drop(['consommationT'],axis= 1)# X=dju_chauffage
#print(X, y)
```

```python
model=LinearRegression() #Création du model de regression lineaire
model.fit(X, y)#Entrainer le modèle sur les données X, y divisées en deux tableaux
model.score(X, y)# Evaluer le modèle
```

```
5]: 0.9417490575514472
```

On obtient un score de 94% c'est le coefficient de détermination est de R²=1-(sum_i=1^n(yi-ŷi)²)/(sum_i=1^n(yi-bar{ŷ}i)²) de la méthode du moindre carrée.

```python
prediction=model.predict(X)#Utiliser le model pour la prédire
figure(figsize=(12, 6))
plt.scatter(X, y)#Nuage de point
plt.plot(conso_dju['dju_chauffage'], prediction, c='r')#La droite de régression
plt.title('Prédiction de la consommation d\'électricité en fonction des DJU')
plt.xlabel('dju_chauffage')
plt.ylabel('consommation d\'électricité')
plt.show()
```



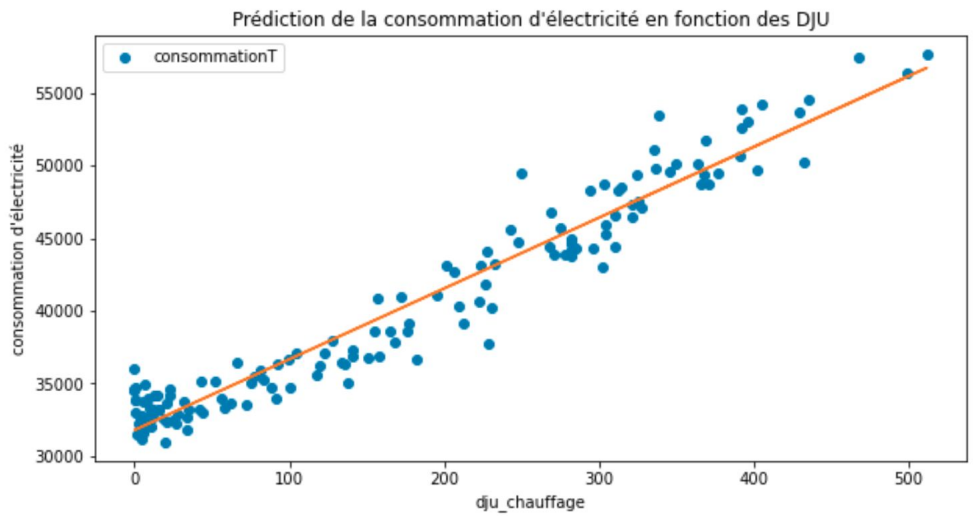
Régression Linéaire avec Statsmodels

```python
#Préparation des données pour établir la Régression linéaire
y = conso_dju['consommationT']
x = sm.add_constant(conso_dju['dju_chauffage'])
```

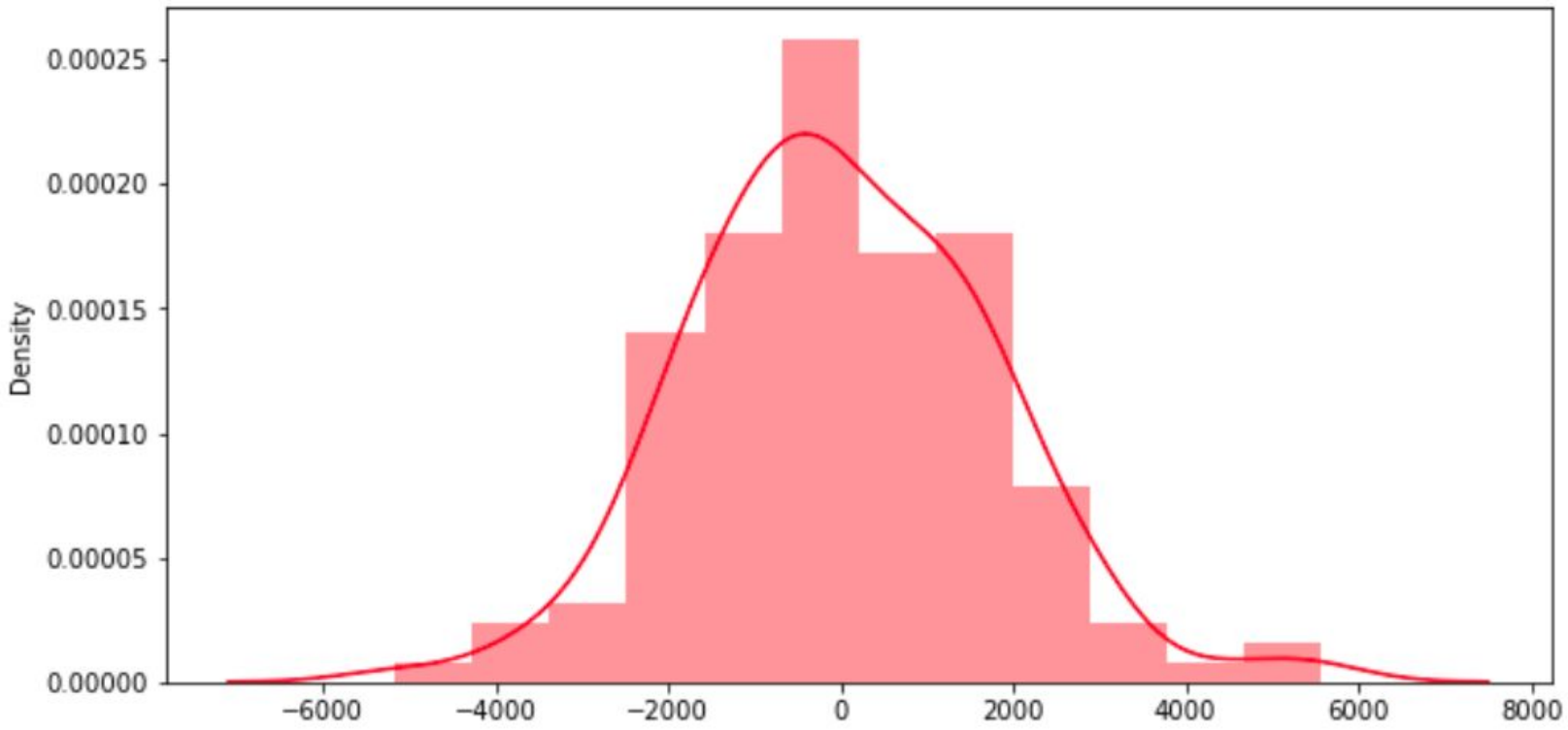
```python
#Régression linéaire Statsmodels
#Definition du modèle et entrainement
reg = sm.OLS(y, x).fit()#(Ordinary Least Squares)
#summary: affichage des caractéristique du modèle de régression construit
print(reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          consommationT   R-squared:                       0.942
Model:                            OLS   Adj. R-squared:                  0.941
Method:                 Least Squares   F-statistic:                     2280.
Date:                Thu, 15 Dec 2022   Prob (F-statistic):           6.21e-89
Time:                        11:59:29   Log-Likelihood:                -1269.3
No. Observations:                 143   AIC:                             2543.
Df Residuals:                     141   BIC:                             2549.
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const           3.18e+04    228.344    139.246      0.000    3.13e+04    3.22e+04
dju_chauffage    48.6790      1.020     47.745      0.000      46.663      50.695
==============================================================================
Omnibus:                        3.141   Durbin-Watson:                   1.555
Prob(Omnibus):                  0.208   Jarque-Bera (JB):                2.828
```

```python
#Représentation de la Régression linéaire obtenue
conso_dju.plot(x='dju_chauffage', y='consommationT', style='o', figsize=(10, 5))
prediction=reg.predict(sm.add_constant(conso_dju['dju_chauffage']))#Pour la prédiction
plt.plot(conso_dju['dju_chauffage'], prediction)
plt.title('Prédiction de la consommation d\'électricité en fonction des DJU')
plt.xlabel('dju_chauffage')
plt.ylabel('consommation d\'électricité')
plt.show()
```



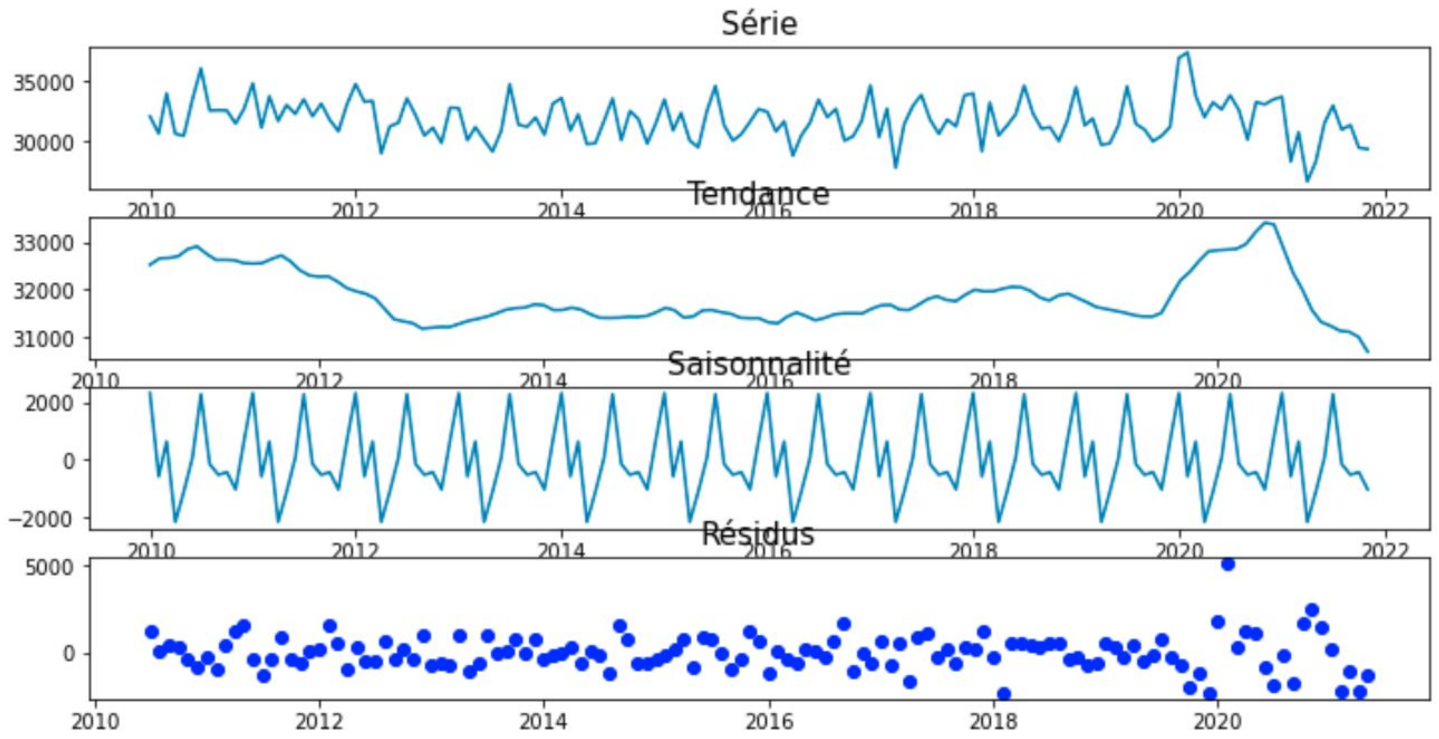
```
#Verification de l'hypothèse de normalité de residu Epsilon= min(y-(ax+b))
figure(figsize=(10, 5))
sns.distplot(reg.resid, color='red')
plt.show()
```



La distribution des résidus est satisfaisante.
En effet les résidus suivent bien la distribution théorique d'une loi normale

Série temporelle avec le modèle additif

```
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt
#On calcul l'ecart saisonnier entre deux pic ou deux creu
decomposition = seasonal_decompose(conso_dju['conso_correction'],  model='additive')

fig, ax = plt.subplots(4,1, figsize=(12,6))
ax[0].plot(decomposition.observed)
ax[0].set_title('Série', fontsize=15)# serie temporelle X(t)
ax[1].plot(decomposition.trend)
ax[1].set_title('Tendance', fontsize=15)#Tandance: levolution générale dans le temps
ax[2].plot(decomposition.seasonal)
ax[2].set_title('Saisonnalité', fontsize=15)#Saisonnalité  S(t)=Y-tandance
ax[3].plot(decomposition.resid,'bo')
ax[3].set_title('Résidus', fontsize=15)#Residu= X(t)-Tendance(t)-S(t)
plt.show()
```



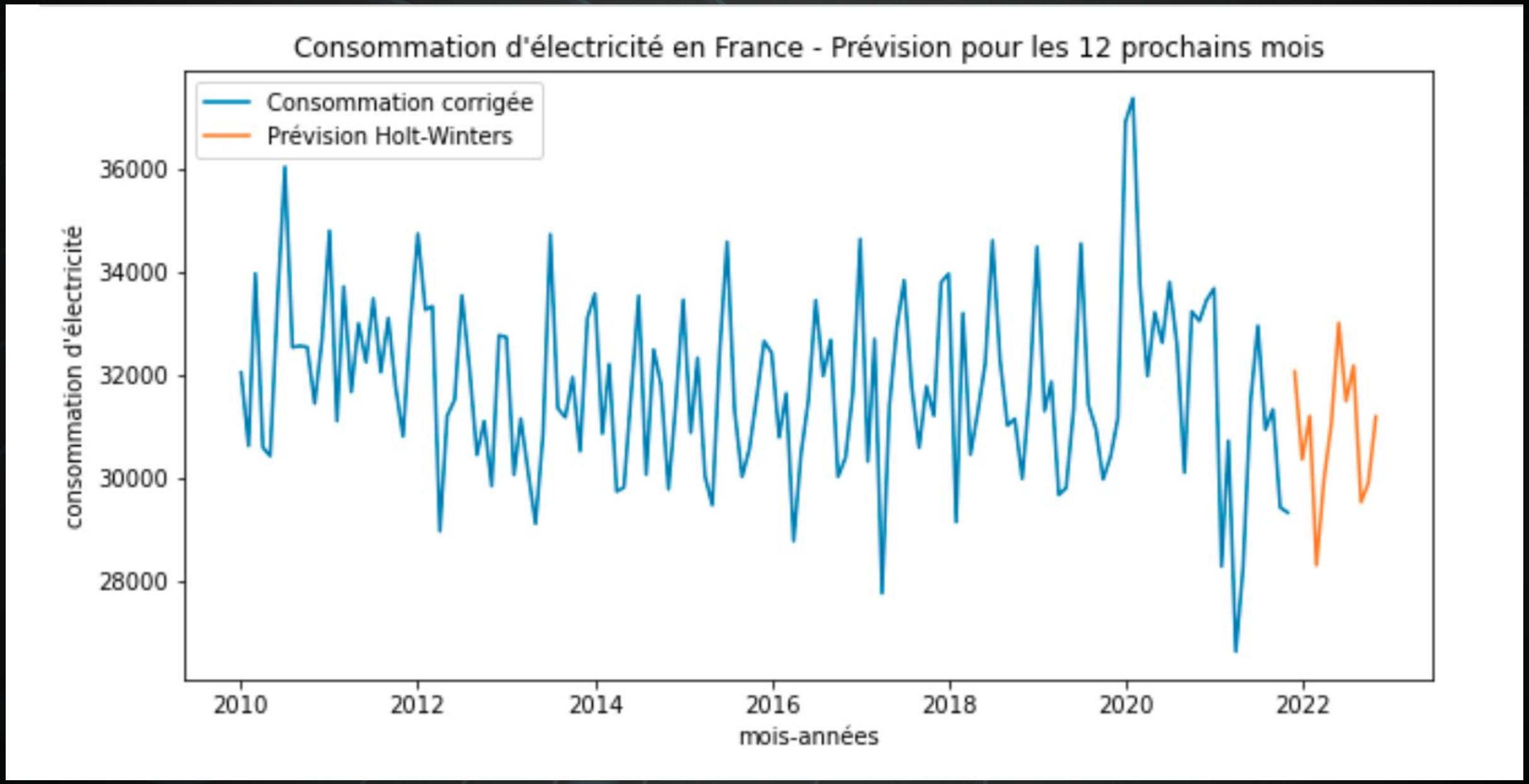
Prévision par lissage exponentiel par la méthode de Holt-Winters

```python
from statsmodels.tsa.api import ExponentialSmoothing

y = np.asarray(conso_dju["conso_correction"])
hw = ExponentialSmoothing(y, seasonal_periods=71, trend='add', seasonal='add').fit()

hw_pred0 = hw.forecast(12)
print(hw_pred0)
```

```
[32061.79135883 30355.71785545 31200.17037064 28306.5429269
 29971.88286687 31035.64330712 33002.66309132 31486.4728327
 32174.97077571 29529.24022186 29897.64965973 31184.2099622 ]
```

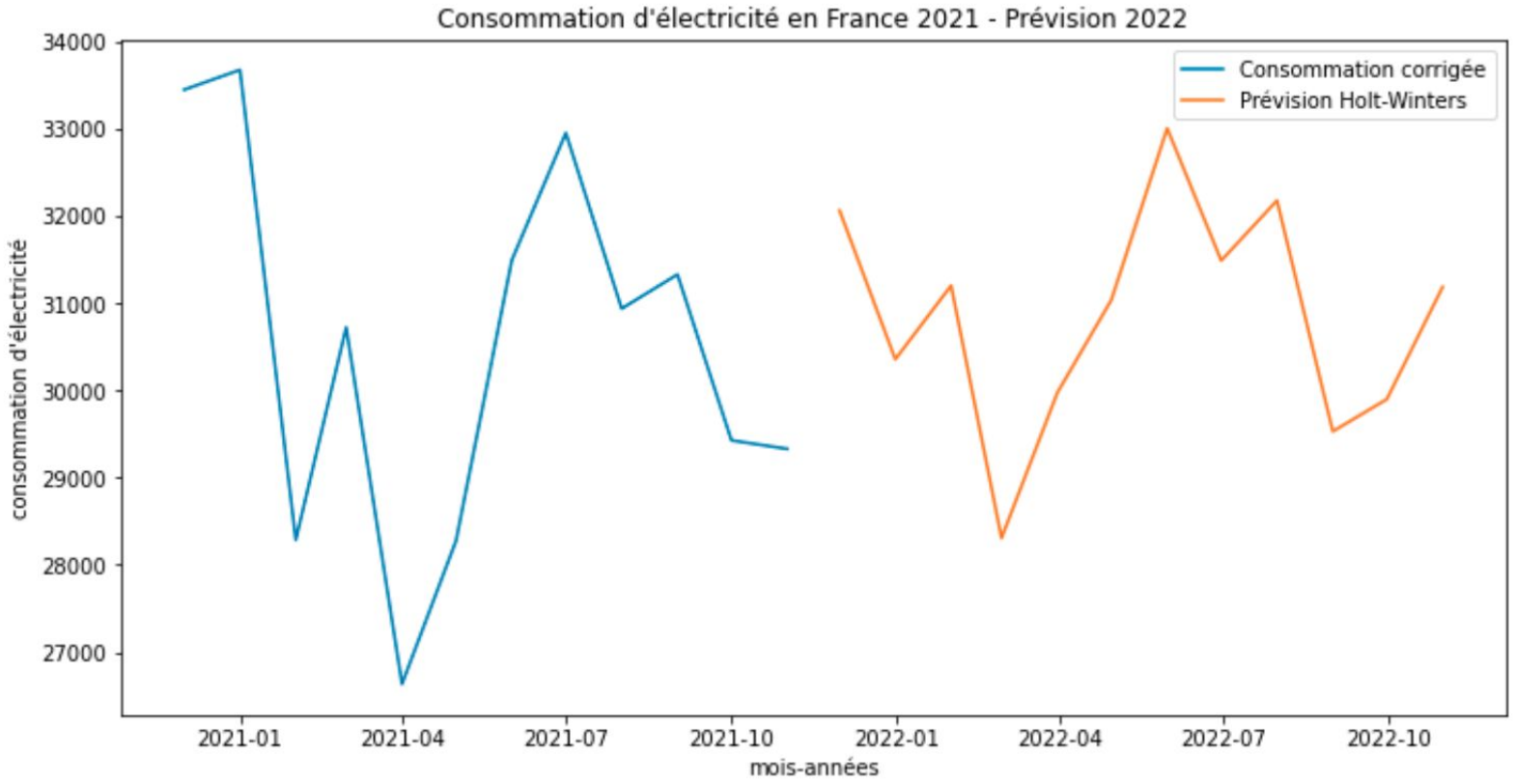
```python
# Visualisation de la prévision sur 12 mois par Holt-Winters
figure(figsize=(12, 6))
plt.plot(conso_dju["conso_correction"], label='Consommation corrigée')
plt.plot(pd.date_range(conso_dju.index[len(y)-1], periods=12, freq='M'), hw_pred0, label='Prévision Holt-Winters'
plt.title("Consommation d'électricité en France - Prévision pour les 12 prochains mois")
plt.xlabel('mois-années')
plt.ylabel('consommation d\'électricité')
plt.legend()
plt.show()
```

Prévision par lissage exponentiel par la méthode de Holt-Winters



Prévision par lissage exponentiel par la méthode de Holt-Winters

```
#Visualisation par restriction sur l'année de prévision et n-1 (2021-2022)
figure(figsize=(12, 6))
plt.plot(conso_dju["conso_correction"].iloc[-12:], label='Consommation corrigée')
plt.plot(pd.date_range(conso_dju.index[len(y)-1], periods=12, freq='M'), hw_pred0, label='Prévision Holt-Winters'

plt.title("Consommation d'électricité en France 2021 - Prévision 2022")
plt.xlabel('mois-années')
plt.ylabel('consommation d\'électricité')
plt.legend()
plt.show()
```



Conclusion

En conclusion, l'utilisation des techniques de machine learning pour analyser, classer et prédire la consommation d'électricité peut offrir de nombreux avantages, tels que l'optimisation de la production d'électricité et la réduction des coûts associés à la consommation d'énergie. Grâce à leur capacité à extraire des informations précieuses à partir de données complexes, les algorithmes d'apprentissage automatique peuvent jouer un rôle crucial dans la gestion efficace de la consommation d'électricité. Cependant, il est important de noter que cette approche doit être utilisée avec prudence et en prenant en compte les différents facteurs qui peuvent affecter la consommation d'électricité, afin de garantir des résultats précis et fiables.