Présentation projet 9

Prédisez la demande en électricité

Partie 1 : Présentation des données

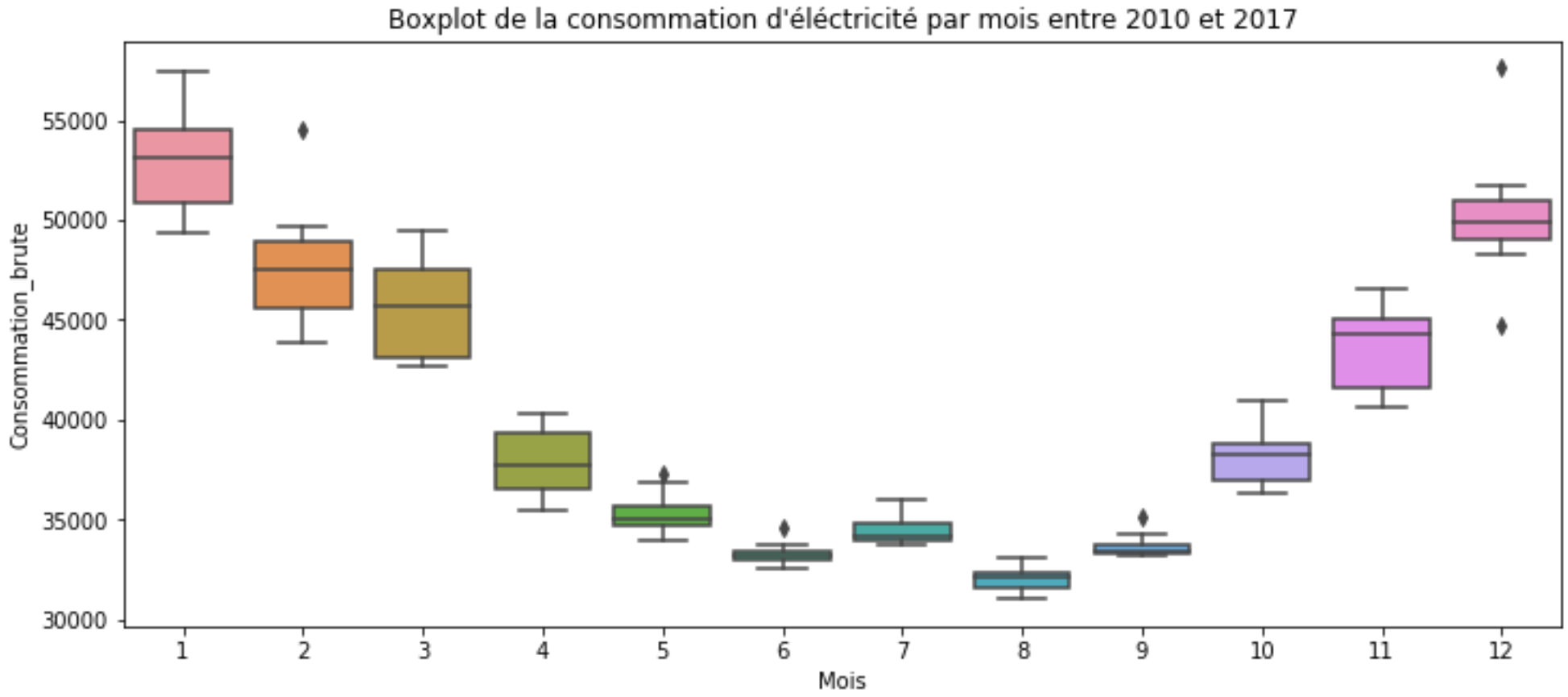
Données de consommation d'électricité mensuelles

Réseau de transport d'électricité

Degré Jour Unifié (DJU)

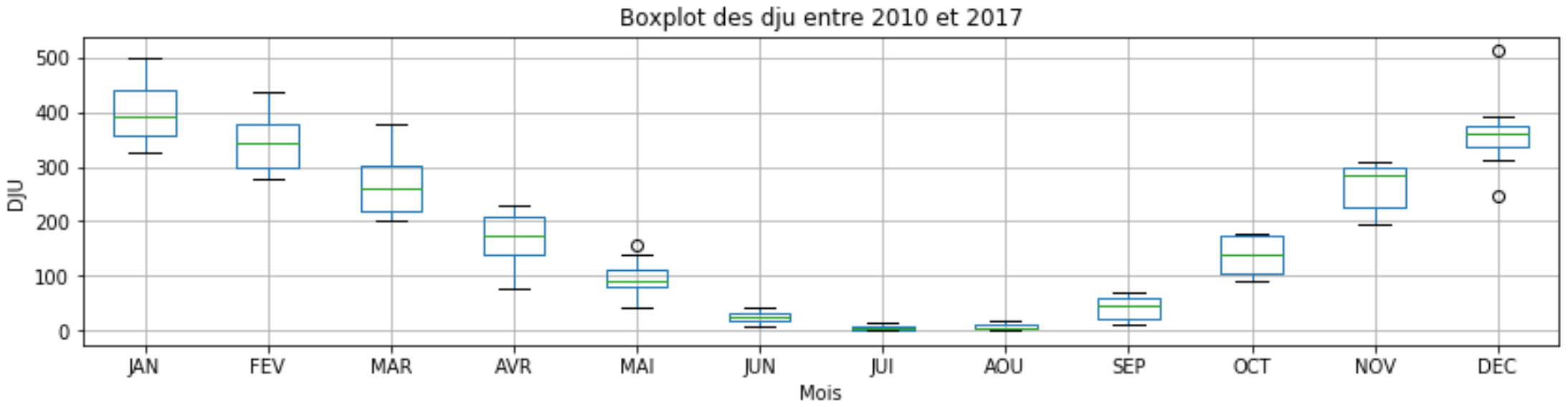
L'expertise efficacité énergétique de GRDF

Partie 1 : Présentation des données



Données de janvier 2010 à décembre 2018.
13 valeurs pour un territoire autre que « France ». Données mises de côté

Partie 1 : Présentation des données



Données jusqu'en mai 2018. Année 2018 mise de côté

Partie 2 : Détails du programme

```python
def plot_sortie_acf( y_acf, y_len, pacf=False):
    "représentation de la sortie ACF"
    if pacf:
        y_acf = y_acf[1:]
    plt.figure(figsize=(20,8))
    plt.bar(range(len(y_acf)), y_acf, width = 0.1)
    plt.xlabel('lag')
    if pacf:
        plt.ylabel('PACF')
        plt.title("PACF Graph")
    else :
        plt.ylabel('ACF')
        plt.title("ACF Graph")
    plt.axhline(y=0, color='black')
    plt.axhline(y=-1.96/np.sqrt(y_len), color='b', linestyle='--', linewidth=0.8)
    plt.axhline(y=1.96/np.sqrt(y_len), color='b', linestyle='--', linewidth=0.8)
    plt.ylim(-1, 1)
    plt.show()
    return
```

Partie 2 : Détails du programme

```python
def difference(dataset, interval):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return pd.Series(diff)
```

Partie 3 : Correction de l'effet température

```
# Régression linéaire
```

```
y = eco.Consommation_brute.copy()
X = df_dju.copy()
X = sm.add_constant(X)
```

```
results=sm.OLS(y,X).fit()
results.params
```

```
const    31831.938004
dju         48.564302
dtype: float64
```

OLS Regression Results

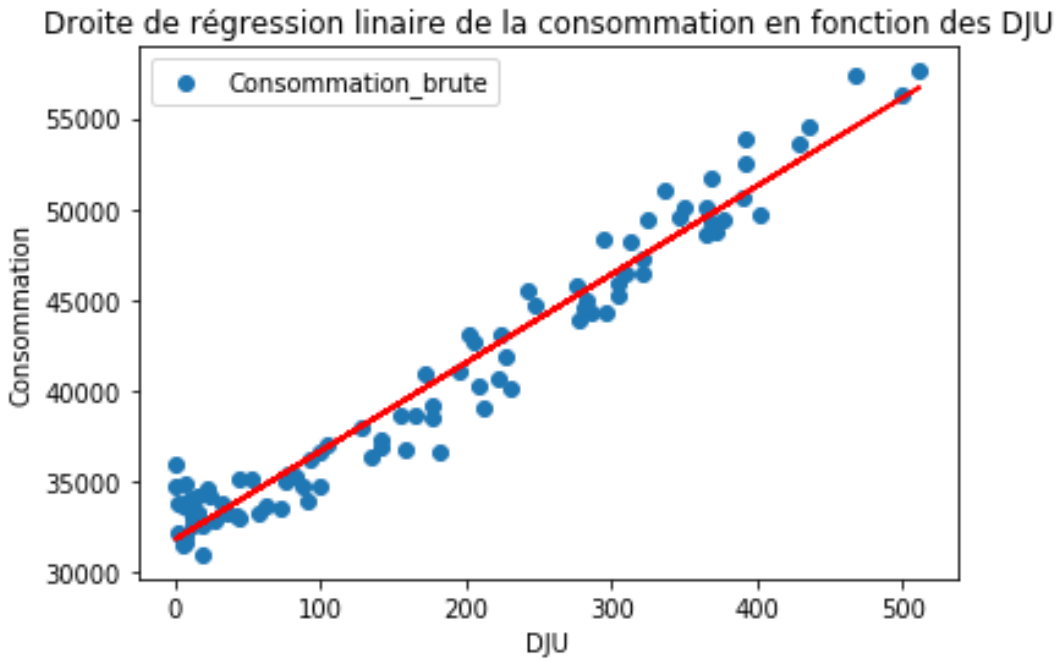
Dep. Variable:	Consommation_brute	**R-squared:**	0.956
Model:	OLS	**Adj. R-squared:**	0.955
Method:	Least Squares	**F-statistic:**	2039.
Date:	Thu, 18 Apr 2019	**Prob (F-statistic):**	1.59e-65
Time:	09:50:46	**Log-Likelihood:**	-839.84
No. Observations:	96	**AIC:**	1684.
Df Residuals:	94	**BIC:**	1689.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.183e+04	247.202	128.769	0.000	3.13e+04	3.23e+04
dju	48.5643	1.076	45.151	0.000	46.429	50.700

Omnibus:	0.217	**Durbin-Watson:**	1.836
Prob(Omnibus):	0.897	**Jarque-Bera (JB):**	0.294
Skew:	0.109	**Prob(JB):**	0.863
Kurtosis:	2.839	**Cond. No.**	361.

Partie 3 : Correction de l'effet température

```
plt.scatter(df_dju,eco.Consommation_brute)
plt.plot(df_dju,results.predict(),color="red")
plt.xlabel("DJU")
plt.ylabel("Consommation")
plt.legend()
plt.title("Droite de régression linaire de la consommation en fonction des DJU")
plt.show()
```



Partie 3 : Correction de l'effet température

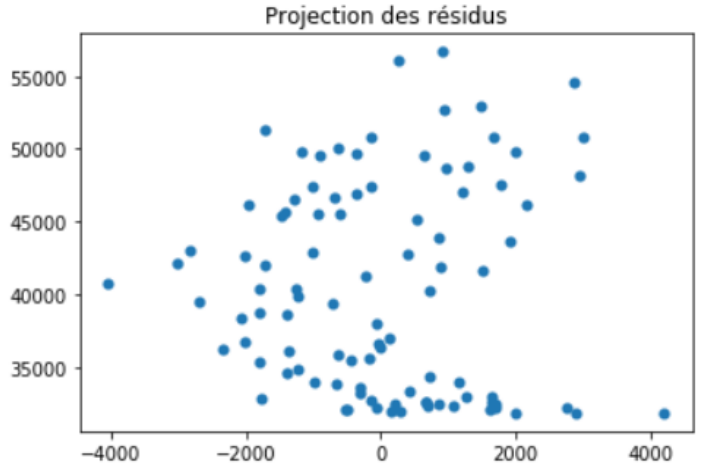
```
#Calcul des résidus
pred_val = results.fittedvalues.copy()
residual = y - pred_val
```

```
# Hypothèse H0 : Il n'y a pas autocorrélation des résidus

sms.durbin_watson(residual)

# Résultat proche de 2. On ne peut donc pas rejetter l'hypothèse
#Il n'y a donc pas autocorrélation des résidus
```

```
1.835666911760647
```

```
fig, ax = plt.subplots()
ax.scatter(residual, pred_val,linewidths=0.1)
plt.title("Projection des résidus")
```

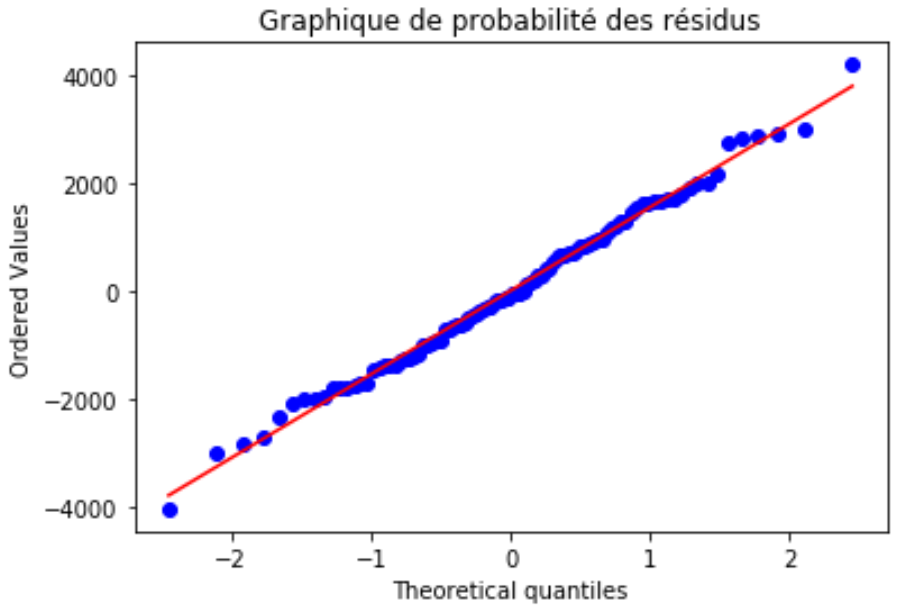
```
Text(0.5, 1.0, 'Projection des résidus')
```




```
# Hypothèse H0 : Les résidus et les les variables exogènes ont la même variance

sms.het_goldfeldquandt(results.resid, results.model.exog)

# La P-value est supérieure au seuil de 5%. L'Homostédasticité est donc acceptée.
```

```
(0.9648797314741909, 0.5479881595271596, 'increasing')
```

Partie 3 : Correction de l'effet température

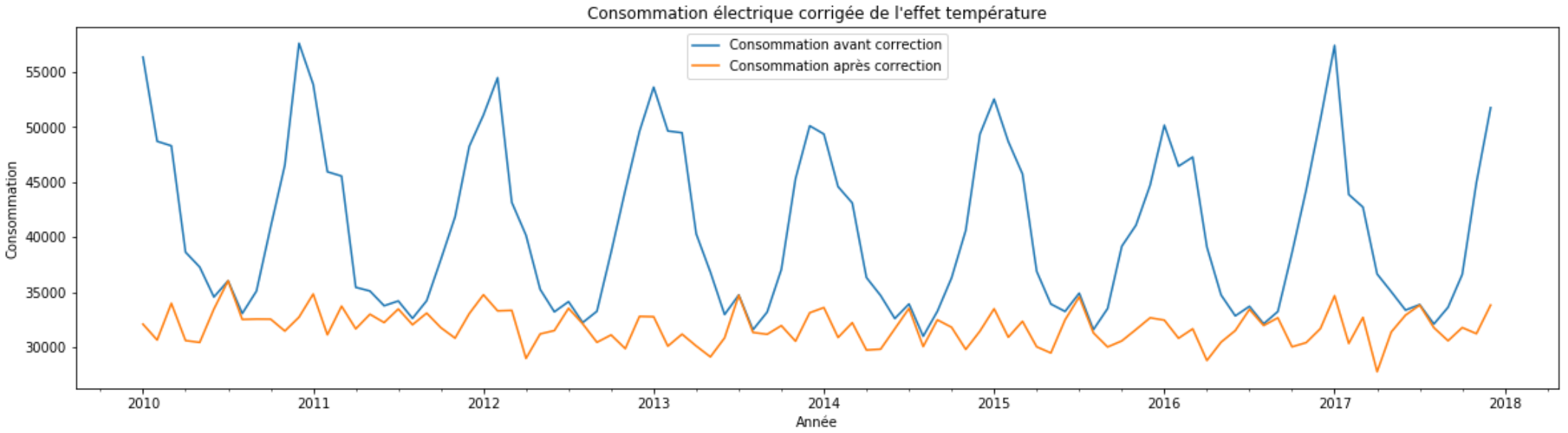



```
#Hypothèse H0 : les résidus suivent une distribution gaussienne

sms.jarque_bera(residual)[:2]

#La p-value est supérieure au seuil de 5%. On ne rejette donc pas l'hypothèse H0.
#La distribution des résidus est gaussienne
```

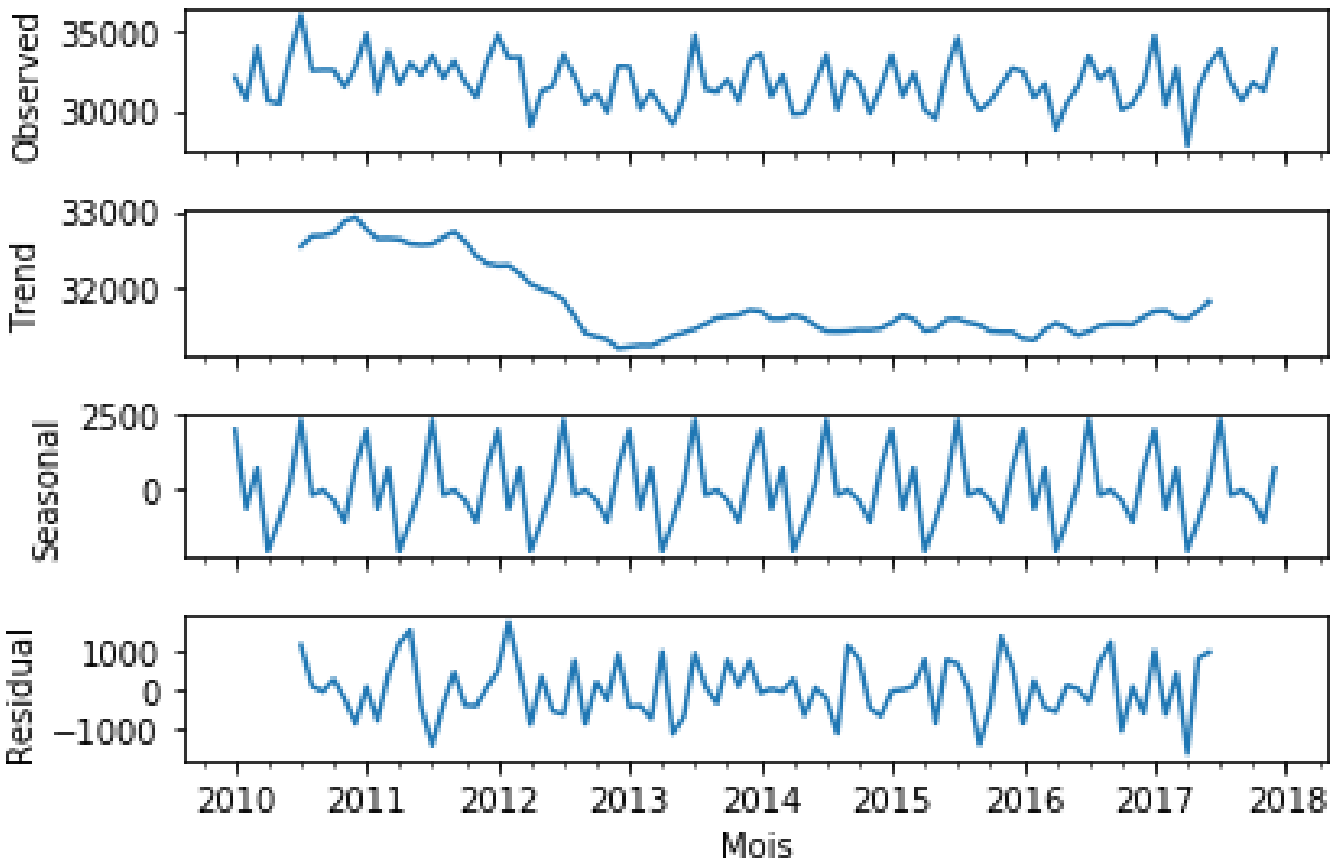
```
(0.2940991425845843, 0.8632511838789051)
```

Partie 3 : Correction de l’effet température

```
pd.merge(eco,conso_corrige,how="left",left_index=True,right_index=True).rename(columns={"Consommation_brute":"Consommation avant
plt.title("Consommation électrique corrigée de l'effet température")
plt.ylabel("Consommation")
plt.xlabel("Année")
plt.show()
#On a enlevé 95% de la variance, on a donc une courbe qui varie peu
```



Partie 4 : Désaisonnalisation



Partie 4 : Désaisonnalisation

```python
def MM12(df):
    i=0
    liste=[]
    df=df.reset_index(drop=True)
    for i in range(df.shape[0]):
        if i<6 :
            liste.append(np.nan)
        elif i>(df.shape[0]-6):
            liste.append(np.nan)
        else :
            liste.append(float(df.loc[(i-6):(i+5)].mean()))
    return(liste)
```

```python
MM=conso_corrige.copy()
MM["Moyenne_mobile"]=MM12(conso_corrige)
MM=MM.drop(columns={"consommation_corrigée"})
```

Partie 4 : Désaisonnalisation

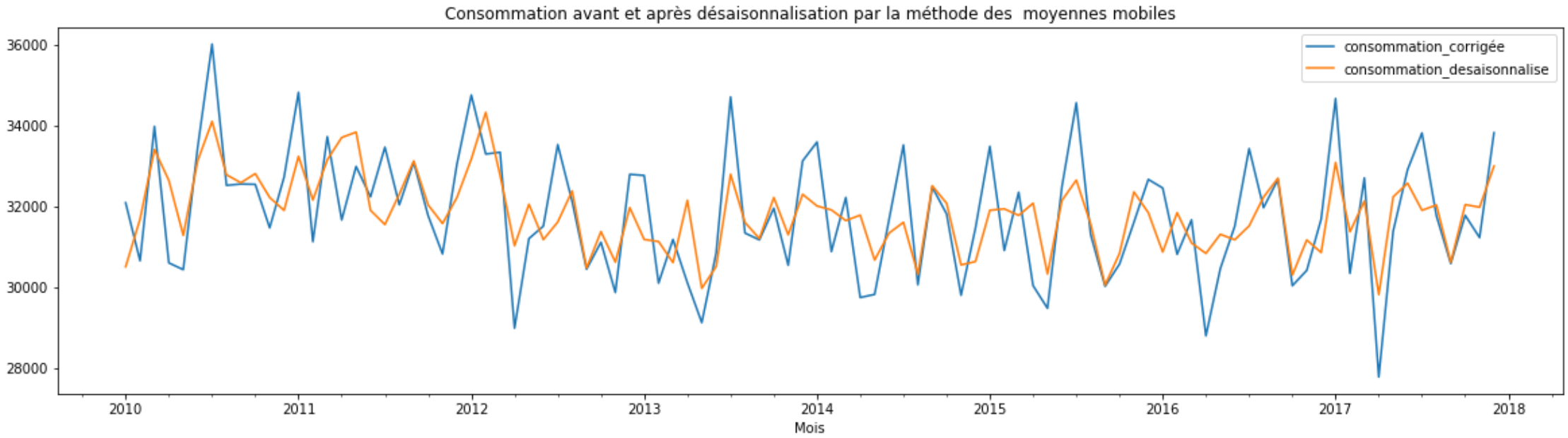
```
#Calculer conso_corrige - Moyenne__Mobile

MECO=(conso_corrige["consommation_corrigée"]-MM["Moyenne_mobile"]).dropna()
MECO=MECO.reset_index()
```

```
# Calcul des coefficients saisonniers

MECO2=MECO.copy()
MECO2["Mois"]=MECO.Mois.dt.month
MECO2["Mois"]=pd.to_numeric(MECO2["Mois"])
MECO3=MECO2.groupby("Mois").mean()
z=MECO3.mean()
```

```
A=[]
for i in MECO3.index :
    A.append(MECO3.loc[i]-z)
```

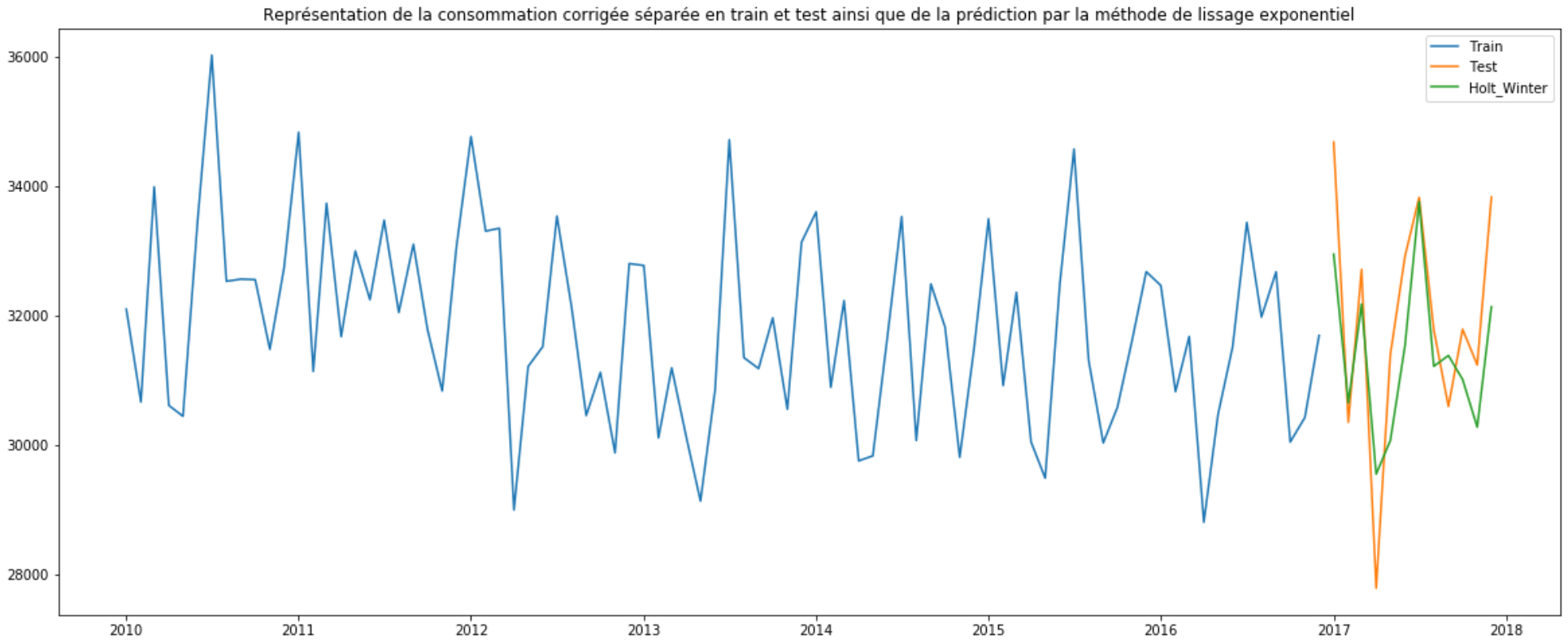
Partie 4 : Désaisonnalisation

```
#Je retranche mes coefficients saisonniers à la série initiale

A=pd.DataFrame(A)
B=[]
for i in range(96) :
    B.append(float(conso_corrige.reset_index().loc[i][1]-A.loc[i%12]))
```



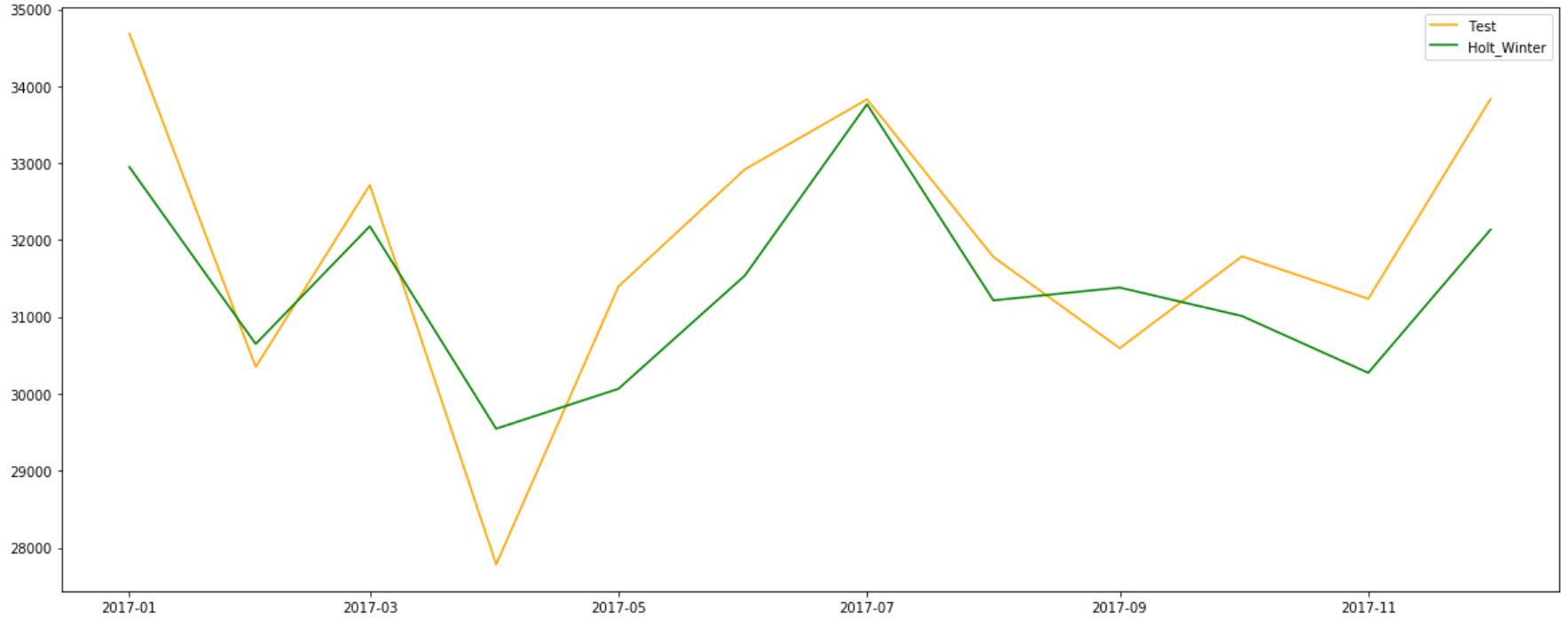
Partie 5 : Holt Winters

```
test = conso_corrige.consommation_corrigée.reset_index()[conso_corrige.consommation_corrigée.reset_index()["Mois"]>"2016-12-01"
y_hat_avg = test.copy()
train = conso_corrige.consommation_corrigée.reset_index()[conso_corrige.consommation_corrigée.reset_index()["Mois"]<"2017-01-01
fit1 = ExponentialSmoothing(np.asarray(train) ,seasonal_periods=12 ,trend='add', seasonal='add',).fit()
y_hat_avg['Holt_Winter'] = fit1.forecast(len(test))
plt.figure(figsize=(20,8))
plt.plot( train, label='Train')
plt.plot(test, label='Test')
plt.plot(y_hat_avg['Holt_Winter'], label='Holt_Winter')
plt.title("Représentation de la consommation corrigée séparée en train et test ainsi que de la prédiction par la méthode de lis
plt.legend(loc='best')
plt.show()
```



Partie 5 : Holt Winters

Comparaison de la prédiction du modèle Holt Winters à la réalité (test)

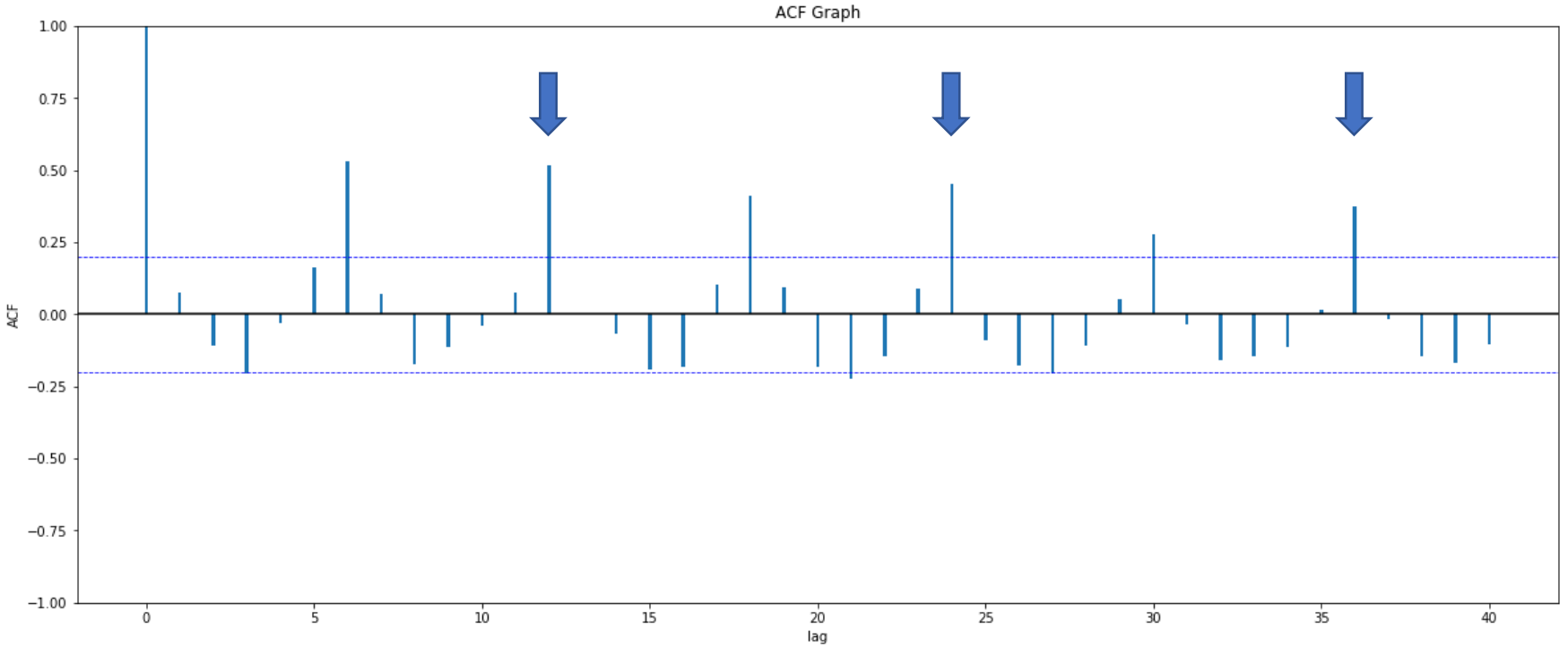



```
# Test de normalité des résidus

# H0 : Les résidus suivent un distribution gaussienne.
st.kstest(HW_residuals,'norm')
# La p-value est au dessus du seuil de 5%, on ne rejette donc pas l'hypothèse H0.
# Les résidus ne suivent donc pas une distribution gaussienne.
```

```
KstestResult(statistic=1.0, pvalue=0.0)
```

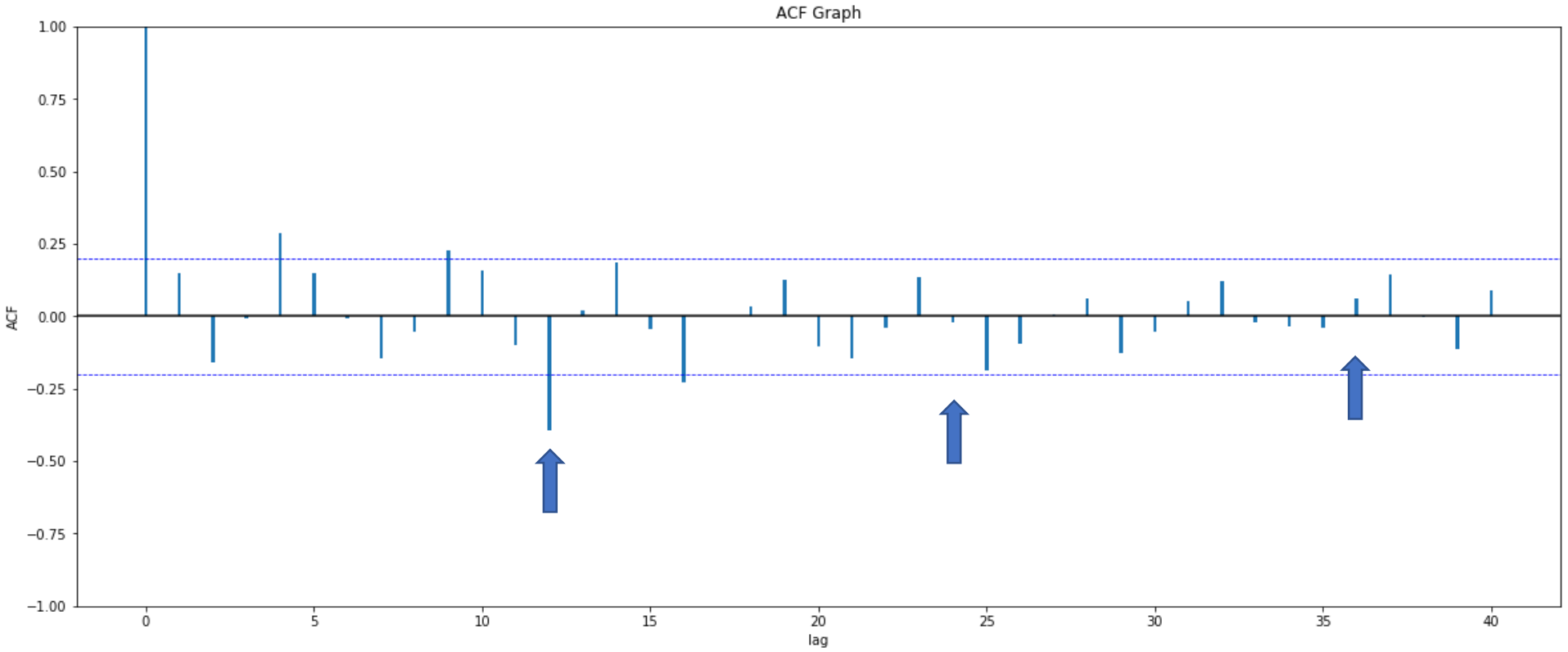
RMSE : 1136,61

Partie 6 : SARIMA



Série non différenciée

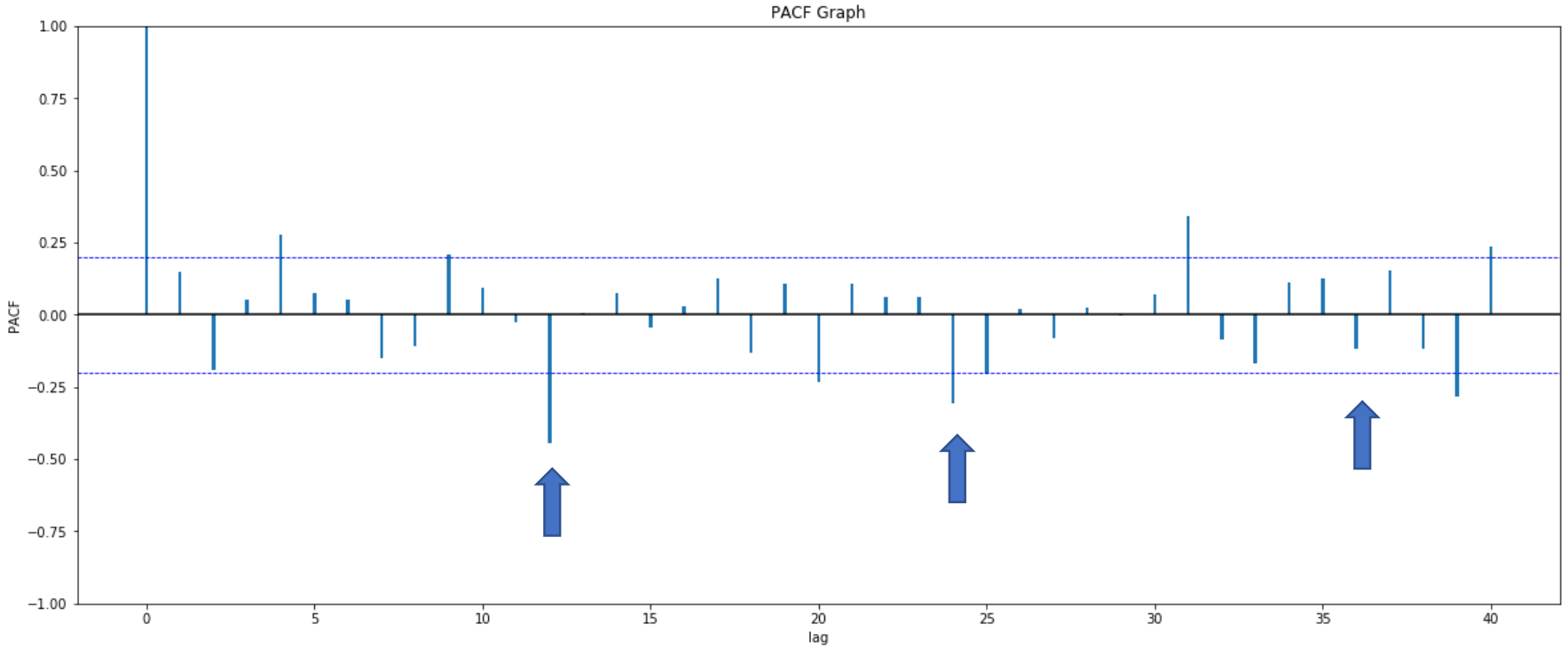
Partie 6 : SARIMA



Différenciation de 12
donc s=12

d=0 D=1
q=1 Q=1

Partie 6 : SARIMA



Différenciation de 12
donc s=12

p=1 P=2
d=0 D=1
q=1 Q=1

Partie 6 : SARIMA

```
model = SARIMAX(np.asarray(conso_corrige["Consommation_corrigée"]), order=(1,0,1), seasonal_order=(2,1,1,12))
results = model.fit()
print(results.summary())
```

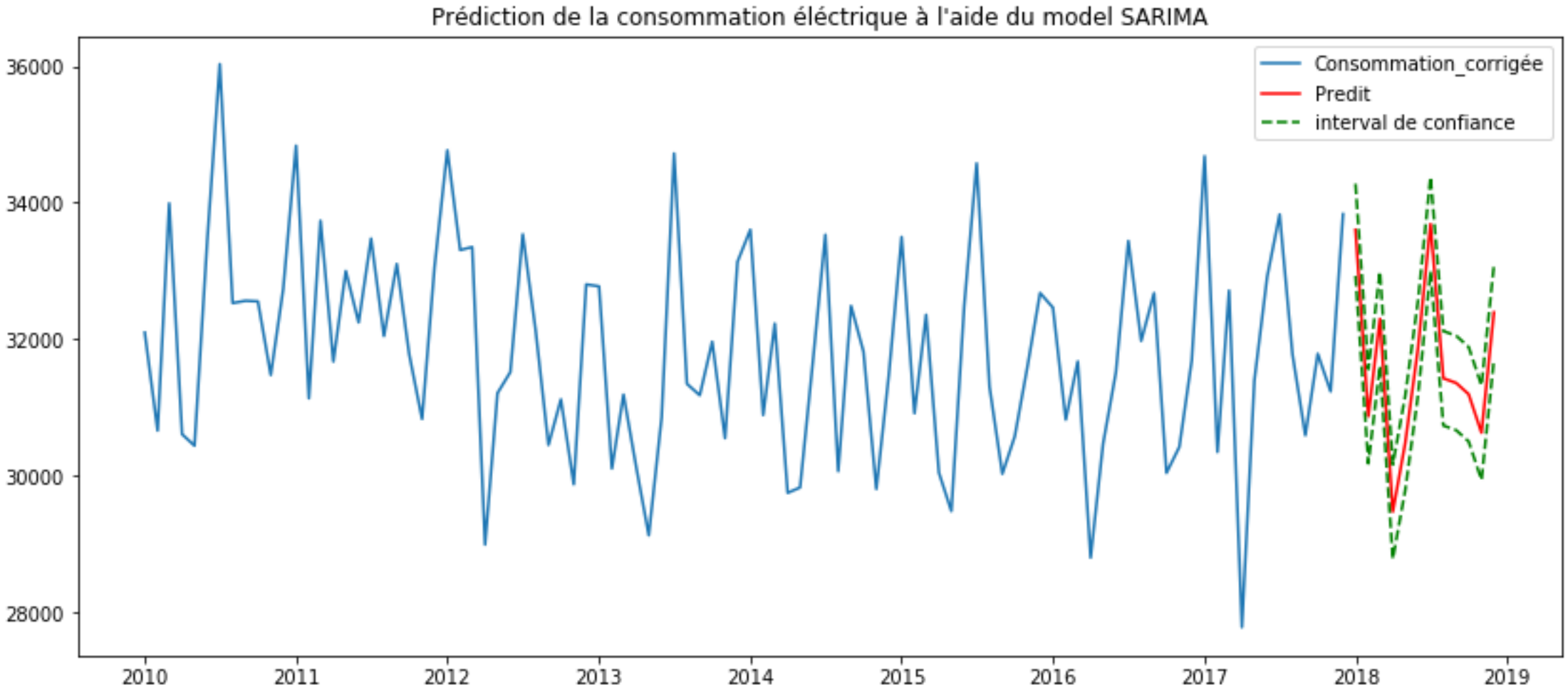
```
                                Statespace Model Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                   96
Model:             SARIMAX(1, 0, 1)x(2, 1, 1, 12)   Log Likelihood                -825.788
Date:                            Thu, 18 Apr 2019   AIC                           1663.576
Time:                                    13:01:55   BIC                           1678.962
Sample:                                         0   HQIC                          1669.795
                                              - 96
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.0000   7.09e-05   1.41e+04      0.000       1.000       1.000
ma.L1         -0.9983      0.034    -28.951      0.000      -1.066      -0.931
ar.S.L12       0.9790      0.328      2.987      0.003       0.337       1.621
ar.S.L24      -0.0167      0.292     -0.057      0.954      -0.588       0.555
ma.S.L12      -0.4563      0.366     -1.247      0.212      -1.174       0.261
sigma2      9.925e+05   5.09e-07   1.95e+12      0.000    9.93e+05    9.93e+05
===================================================================================
Ljung-Box (Q):                       55.81   Jarque-Bera (JB):                 1.11
Prob(Q):                              0.05   Prob(JB):                         0.57
Heteroskedasticity (H):               0.73   Skew:                             0.25
Prob(H) (two-sided):                  0.38   Kurtosis:                         3.17
===================================================================================
```

Partie 6 : SARIMA

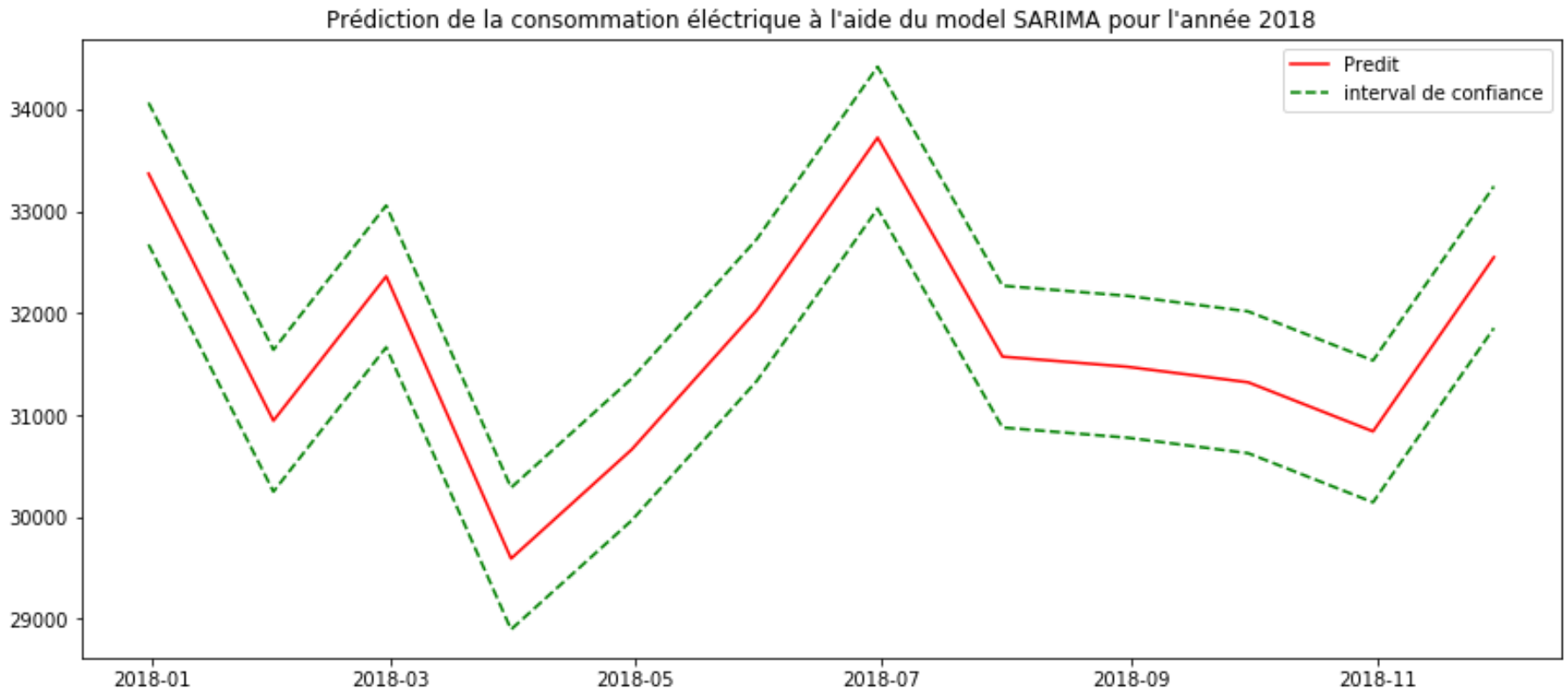
```
model3 = SARIMAX(np.asarray(conso_corrige["Consommation_corrigée"]), order=(1,0,1), seasonal_order=(1,1,1,12))
results3 = model3.fit()
print(results3.summary())
```

```
                                 Statespace Model Results
==========================================================================================
 Dep. Variable:                                  y    No. Observations:                   96
 Model:             SARIMAX(1, 0, 1)x(1, 1, 1, 12)    Log Likelihood                -816.707
 Date:                            Thu, 18 Apr 2019    AIC                           1643.415
 Time:                                    12:59:44    BIC                           1656.236
 Sample:                                         0    HQIC                          1648.597
                                               - 96
 Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          1.0000   6.76e-06   1.48e+05      0.000       1.000       1.000
ma.L1         -0.9953      0.017    -58.617      0.000      -1.029      -0.962
ar.S.L12       0.9816      0.040     24.579      0.000       0.903       1.060
ma.S.L12      -0.7933      0.231     -3.430      0.001      -1.247      -0.340
sigma2      1.056e+06   2.24e-07   4.72e+12      0.000    1.06e+06    1.06e+06
===================================================================================
Ljung-Box (Q):                       59.67   Jarque-Bera (JB):                 1.31
Prob(Q):                              0.02   Prob(JB):                         0.52
Heteroskedasticity (H):               0.81   Skew:                             0.28
Prob(H) (two-sided):                  0.56   Kurtosis:                         3.07
===================================================================================
```

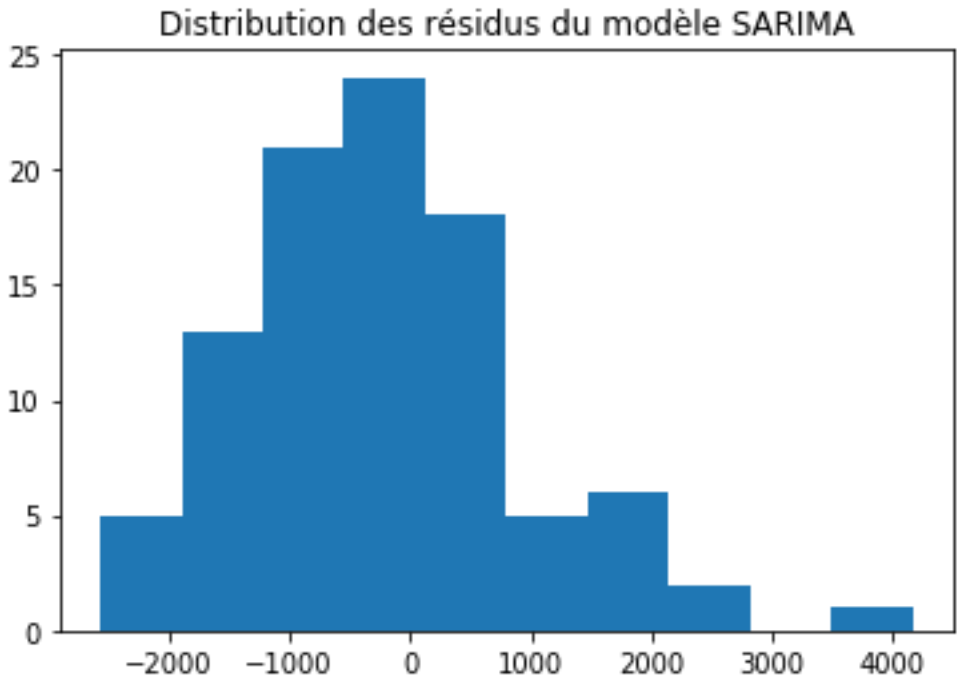
Partie 6 : SARIMA



Partie 6 : SARIMA



Partie 6 : SARIMA

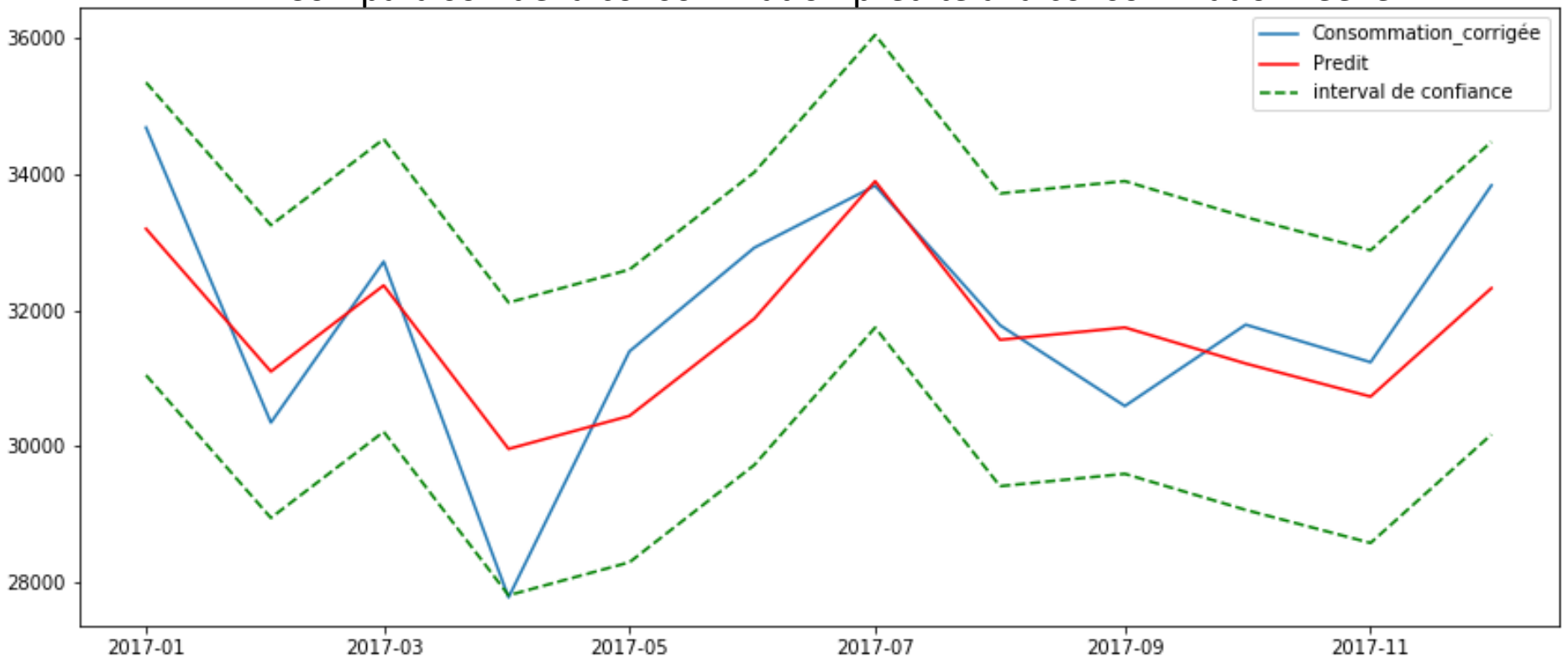



```
# Test de normalité des résidus

# H0 : Les résidus suivent un distribution gaussienne.
st.kstest(residuals,'norm')
# La p-value est inférieure du seuil de 5%, on rejette donc l'hypothèse H0.
# Les résidus ne suivent donc pas une distribution gaussienne.
```

```
KstestResult(statistic=1.0, pvalue=0.0)
```

Partie 6 : SARIMA

Comparaison de la consommation prédite à la consommation réelle



RMSE : 1075,47