المعهد الوطني للبريد والمواصلات
ⵓⵙⴷⵉⵡ ⵓⵏⴰⵎⵓⵙ ⵏ ⵜⵙⵙⵓⵏⴼⵉ ⴷ ⵜⵙⵡⵉⵍⵜ
Institut National des Postes et Télécommunications

Mini-projet

LANGAGE PYTHON AVANCÉ

Etude (*Prévoir*)d'une Série Chronologique

La série de **la température minimale quotidienne**

Data Engineer

Réalisé par :

Youssef Jehbali

Khalil Mohamed Mouadi

Younes Nibgourine

Encadré par :

Pr.H.Laanaya

Année universitaire : 2021-2022

Introduction

Une série chronologique (SC), ou série temporelle, est une série d'observations ordonnées chronologiquement. Elles se rencontrent naturellement dans une grande variété de domaines. On peut citer : l'économie (taux de chômage, . . .), la finance (cours d'action, taux d'intérêt, . . .), l'écologie (pollution à l'ozone, au CO, . . .), le transport (avec l'exemple célèbre du trafic aérien international), la démographie. . .

- On étudie une suite de couples de la forme (t, X_t), où ***Xt*** *est l'observation de la variable à l'instant t (qui peut être le mois, l'année . . .).*

Définitions et concepts

Qu'est-ce qu'une série chronologique ?

- Ensemble d'observations qui se distinguent par le rôle particulier que joue l'ordre (la chronologie) dans lequel elles ont été recueillies : ce sont des observations effectuées à intervalle fixé ***h***, au temps $t_1, t_2, \ldots, t_N$.

Une série chronologique sera notée

$$X(t_1), X(t_2), \ldots, X(t_N)$$

Ou plus simplement

$$X_1, X_2, \ldots\ldots, X_N.$$

Objectifs

L'étude d'une SC permet d'analyser, de décrire et d'expliquer un phénomène au cours du temps et d'en tirer des conséquences pour des prises de décision. L'un des objectifs principaux de l'étude d'une SC est la prévision qui consiste à prévoir les valeurs futures $X_{T+h}(h = 1, 2, 3, \ldots)$ de la série chronologique à partir de ses valeurs observées jusqu'au temps T.

Objectif de Notre Mini-Projet

Dans ce mini projet, On va appliquer la méthode de **Box et Jenkins** sur la série de ***la température minimale quotidienne***

- Il s'agit de construire le modèle adéquat permettant de prévoir la série chronologique. Tout en utilisant les ***packages*** étudiées précédemment de **Python**

Pourquoi Python ?!

La data science est chargée d'analyser, de transformer les données et d'extraire des informations utiles pour la prise de décision. Et il n'y a pas besoin d'avoir des connaissances avancées en programmation pour utiliser ***Python*** afin d'effectuer ces tâches. La programmation et la visualisation des résultats sont plus simples. Il y a peu de lignes de code en Python et ses interfaces graphiques de programmation sont conviviales.

Dans le développement d'un projet de science des données, il existe différentes tâches pour terminer ledit projet, dont les plus pertinentes sont l'extraction de données, le traitement de l'information, le développement d'algorithmes (machine Learning) et l'évaluation des résultats.

❖ DataSetes

Notre Datasets se forme d'un fichier csv qui contient une série chronologique de la température minimale quotidienne

Lien de Datasets :

https://raw.githubusercontent.com/jbrownlee/Datasets/master/daily-min-temperatures.csv

```
"Date","Temp"
"1981-01-01",20.7
"1981-01-02",17.9
"1981-01-03",18.8
"1981-01-04",14.6
"1981-01-05",15.8
"1981-01-06",15.8
"1981-01-07",15.8
"1981-01-08",17.4
```

❖ Bibliothèques nécessaires pour faire l'étude :

```
from dateutil.parser import parse
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller, kpss
!pip install pmdarima
import pmdarima as pm
```

- ***Parse*** est un module qui offre un analyseur générique de chaîne de date/heure qui est capable d'analyser la plupart des formats connus pour représenter une date et/ou une heure.
- ***Matplotlib*** est une bibliothèque complète pour créer des visualisations statiques, animées et interactives en Python. Elle rend les choses faciles faciles et les choses difficiles possibles.
- ***Seaborn*** est une bibliothèque Python de visualisation de données basée sur matplotlib . Il fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs.
- ***NumPy*** est une bibliothèque Python utilisée pour travailler avec des tableaux (le domaine de l'algèbre linéaire, de la transformée de Fourier et des matrices...)

- ***Pandas*** est un outil d'analyse et de manipulation de données open source rapide, puissant, flexible et facile à utiliser,
construit sur le langage de programmation Python .
- ***statsmodels.tsa.stattools.acf :*** pour calculer la fonction d'autocorrélation.
- ***statsmodels.tsa.stattools.pacf :*** pour l'estimation d'autocorrélation partielle
- ***statsmodels.graphics.tsaplots*** : pour tracer les fonctions des deux commandes *d'acf* et de *pacf*.
- ***statsmodels.tsa.seasonal.seasonal_decompose :*** pour la décomposition saisonnière à l'aide de moyennes mobiles.
- ***statsmodels.tsa.stattools.kpss :*** pour Calcule le test de Kwiatkowski-Phillips-Schmidt-Shin (KPSS) pour l'hypothèse nulle selon laquelle x est stationnaire en niveau ou en tendance.
- ***statsmodels.tsa.stattools.adfuller :*** pour le Test de racine unitaire Dickey-Fuller augmenté.
- ***Pmdarima*** est une bibliothèque statistique conçue pour combler le vide dans les capacités d'analyse de séries chronologiques de Python
- ***statsmodels*** est un package Python qui fournit un complément à scipy pour les calculs statistiques, y compris les statistiques descriptives et l'estimation et l'inférence pour les modèles statistiques.

❖ Importation du fichier csv qui contient notre donnée :

```
R_csv= pd.read_csv('https://raw.githubusercontent.com/jbrownlee/Datasets/master/daily-min-temperatures.csv',index_col='Date')
```

Traitement juste des 2000 premières valeurs du fichier csv :

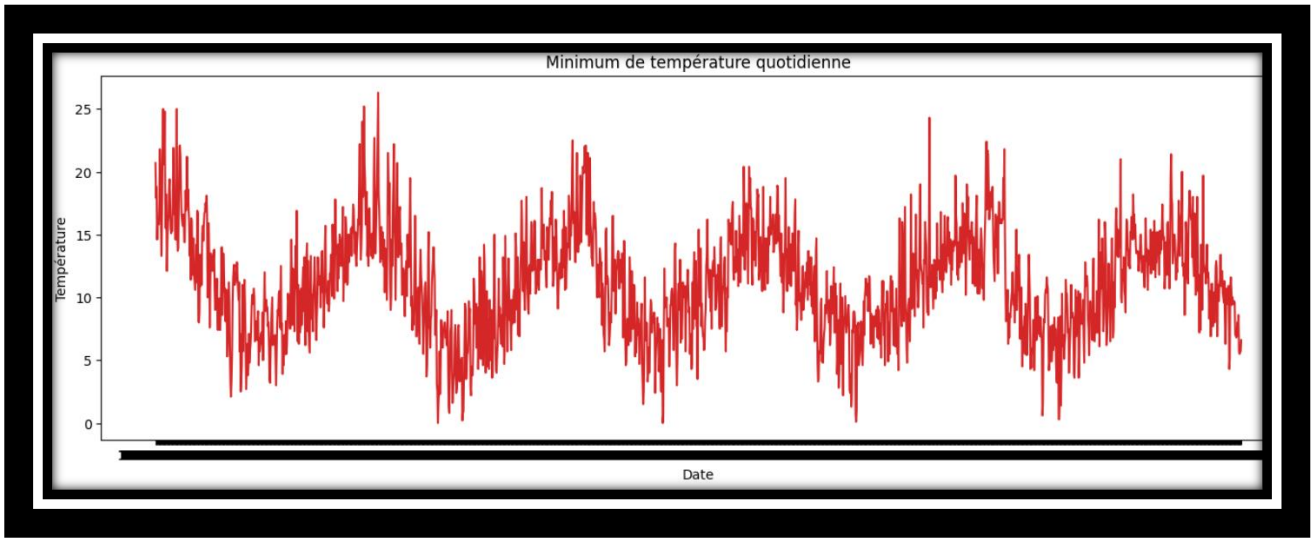
```
data=pd.DataFrame(R_csv['Temp'])
data=data.head(2000)
```

Fractionnement des données en un échantillon d'apprentissage (90%) et un échantillon de validation (10%)

```
train=data[:1800]
test=data[-200:]
```

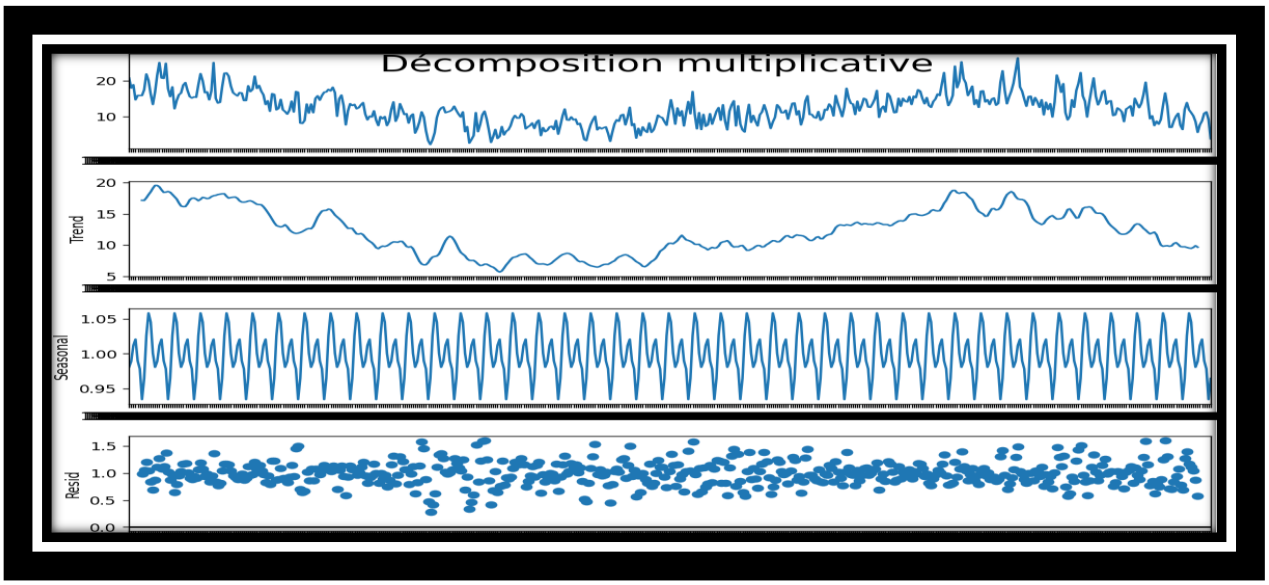
Représentation graphique de 100% de la série

```
plt.figure(figsize=(16,5), dpi=100)
plt.plot(data.index, data.Temp, color='tab:red')
plt.gca().set(title='Minimum de température quotidienne', xlabel='Date', ylabel='Température')
plt.show()
```



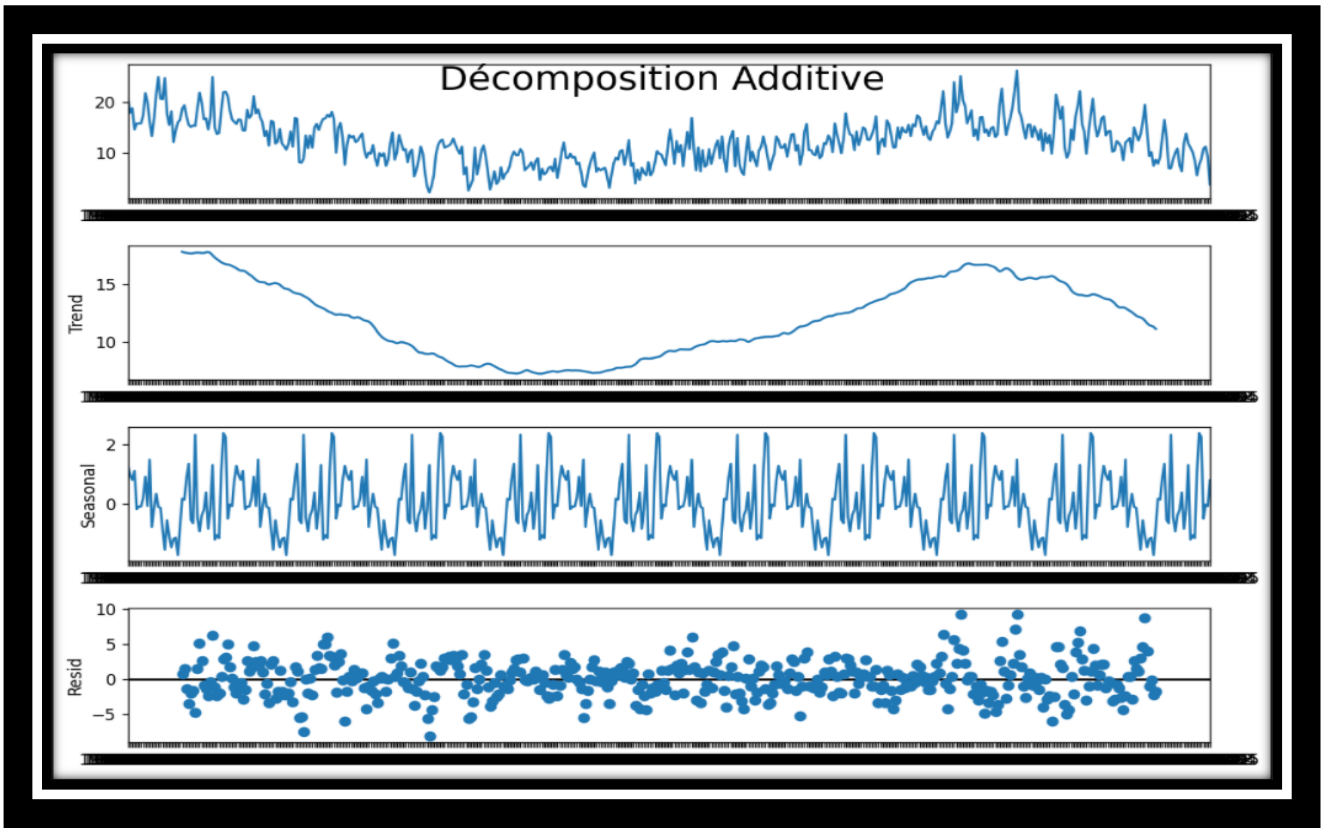
❖ On voit une saisonnalité dans la série on fait ***une décomposition multiplicative*** pour la récupérer

```
result_mul = seasonal_decompose(data.head(500), model='multiplicative',period=12)
result_mul.plot().suptitle('Décomposition multiplicative', fontsize=22)
plt.show()
```



❖ On peut aussi faire ***une décomposition additive***

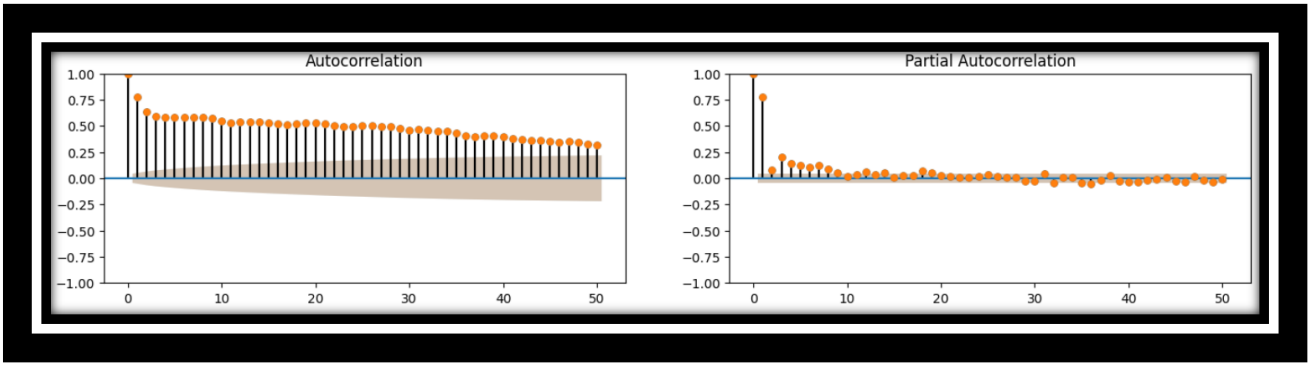
```
result_add = seasonal_decompose(data.head(500), model='additive',period=50)
result_add.plot().suptitle('Décomposition Additive', fontsize=22)
plt.show()
```



O Donc on a bien une **saisonnalité** et **pas** de **tendances**

On trace la fonction d'autocorrélation et la fonction d'autocorrélation partielle pour un décalage d'ordre 50

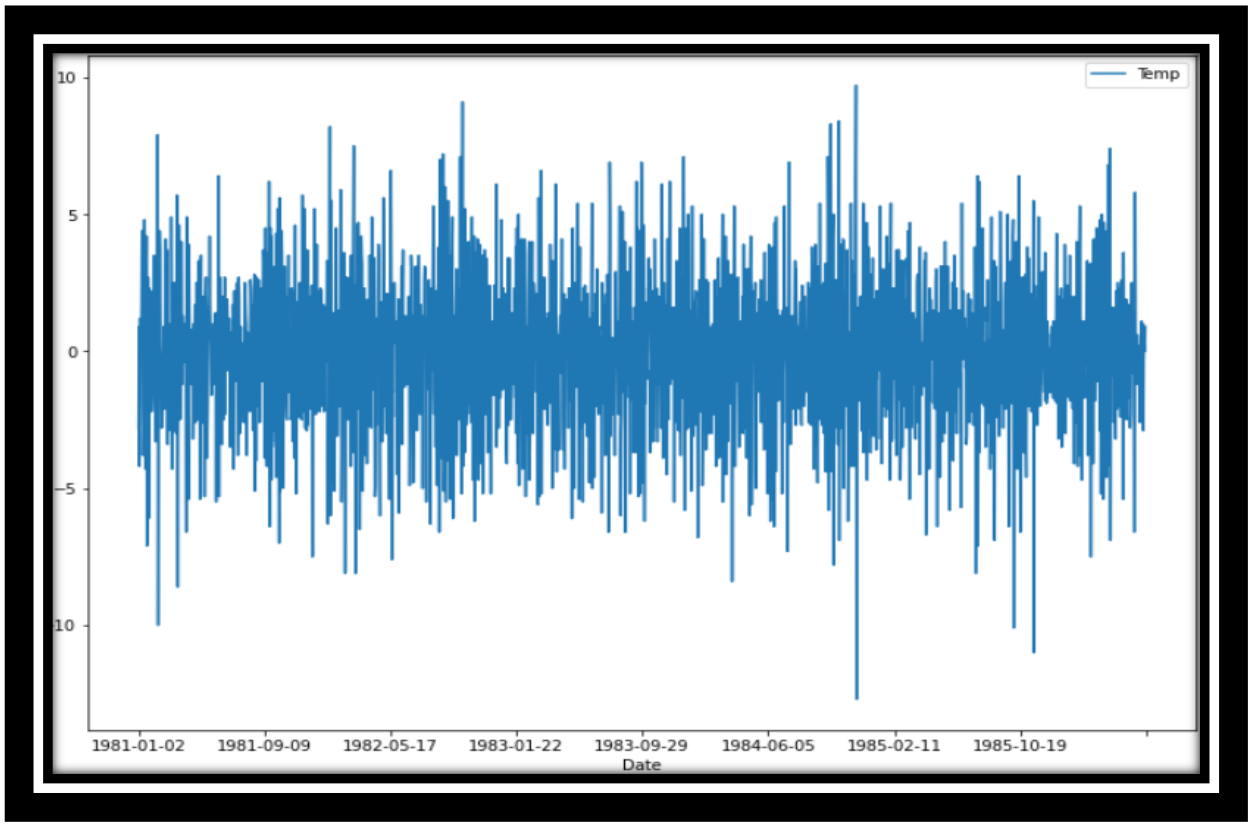
```
fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)
plot_acf(data.Temp.tolist(), lags=50, ax=axes[0])
plot_pacf(data.Temp.tolist(), lags=50, ax=axes[1])
```



La fonction d'autocorrélation ne converge pas vers 0 rapidement donc il faut différencier notre série pour qu'elle devienne stationnaire

```
data_diff = data.diff()
data_diff.dropna(inplace=True)

# Plot de data différencié
fig, ax = plt.subplots(figsize=(12,9))
fig.suptitle('Line Plot of the Stationary Seasonal Time Series Data')
data_diff.plot(ax=ax)
plt.show()
```



On effectue un test de Dickey Fuller augmenté (**ADF**)

```
# ADF Test
result = adfuller(data_diff.Temp.values, autolag='AIC')
print(f'ADF Statistic: {result[0]}')
print(f'p-value: {result[1]}')
for key, value in result[4].items():
    print('Critial Values:')
    print(f'   {key}, {value}')
```

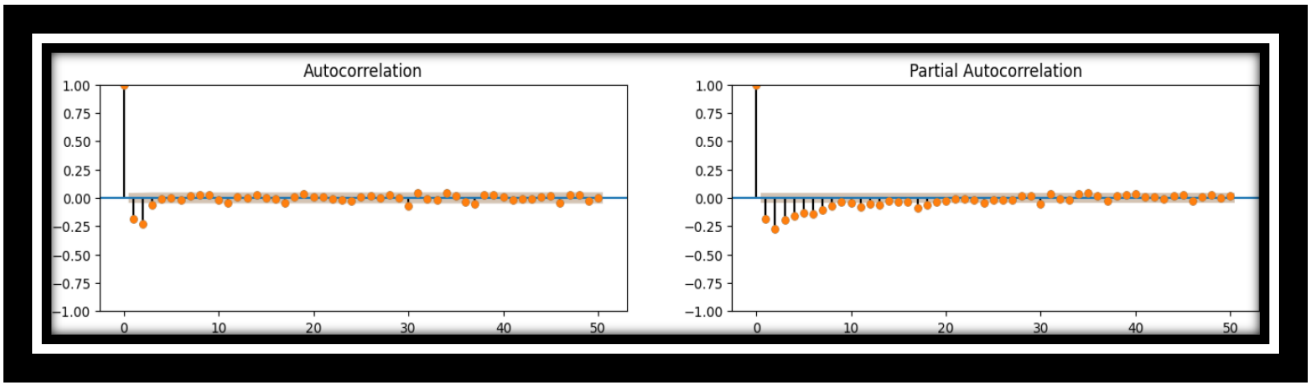
JB

```
ADF Statistic: -16.064496705625658
p-value: 5.592530254605227e-29
Critial Values:
   1%, -3.433656968706682
Critial Values:
   5%, -2.863000832624789
Critial Values:
   10%, -2.567547686205489
```

- P-value est très petit que 5% donc on rejette H0 d'où la série est désormais stationnaire

Représentons l'ACF et la PACF de la série différenciée

```
fig, axes = plt.subplots(1,2,figsize=(16,3), dpi= 100)
plot_acf(data_diff.Temp.tolist(), lags=50, ax=axes[0])
plot_pacf(data_diff.Temp.tolist(), lags=50, ax=axes[1])
```



Convergence rapide vers 0, c'est ce qu'on veut pour appliquer **la méthode de Box-Jenkins**

Trouvons le meilleur model **SARIMA** pour notre série (***car on a une composante saisonniaire « seasonal=True »***)

```
model = pm.auto_arima(train.Temp, d=1,
                      m=12, trend='c', seasonal=True,
                      start_p=0, start_q=0, max_order=6, test='adf',
                      stepwise=True, trace=True)
```

D'où le résultat :

```
  ARIMA(1,1,2)(0,0,0)[12] intercept   : AIC=8411.963, Time=0.84 sec
  ARIMA(3,1,0)(0,0,0)[12] intercept   : AIC=8599.644, Time=0.48 sec
  ARIMA(3,1,2)(0,0,0)[12] intercept   : AIC=8414.584, Time=2.13 sec
  ARIMA(2,1,1)(0,0,0)[12]             : AIC=8411.655, Time=0.82 sec

 Best model:  ARIMA(2,1,1)(0,0,0)[12]
```

Appliquons-le model sur la série

```
sar =sm.tsa.statespace.SARIMAX(train.Temp,order=(2,1,1),seasonal_order=(0,0,0,12))
results = sar.fit()
print("Results of SARIMAX on train")
results.summary().tables[1]
```

Results of SARIMAX on train

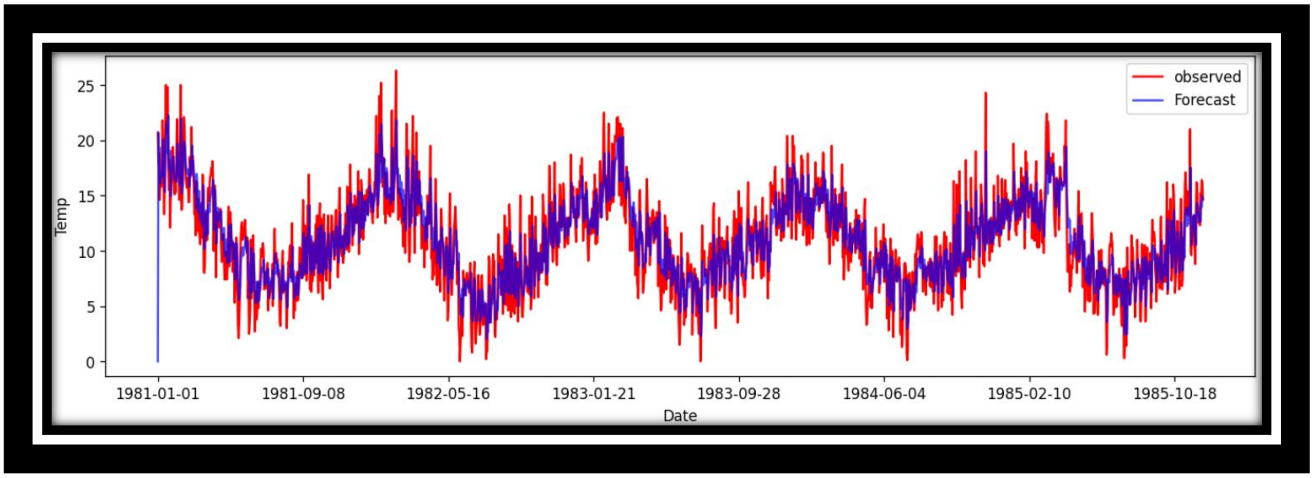
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.4953	0.025	19.738	0.000	0.446	0.544
ar.L2	-0.1365	0.025	-5.500	0.000	-0.185	-0.088
ma.L1	-0.8940	0.014	-62.871	0.000	-0.922	-0.866
sigma2	6.2448	0.200	31.213	0.000	5.853	6.637

- La p-value de tous les coefficients est très petit ce qui est signe d'un bon model

Comparons-le model avec la série originale :

```
pred = results.get_prediction(start='1981-01-01', dynamic=False)
pred_ci = pred.conf_int()
ax = train['Temp'].plot(label='observed',color='r')
pred.predicted_mean.plot(ax=ax, label='Forecast',color='b', alpha=.7, figsize=(14, 4))

ax.set_xlabel('Date')
ax.set_ylabel('Temp')
plt.legend()
plt.show()
```



➢ **Depuis le graphe on voit que notre modèle est le meilleur**