Projet SERIE CHRONOLOGIQUE :

Prédiction des ventes d'un produit

Réalisé par : Lahrichi Youssef

Filière : Data Science

Sommaire

Introduction

Dans un contexte économique en constante évolution, l'analyse des ventes constitue un levier stratégique essentiel pour les entreprises souhaitant anticiper la demande, ajuster leur production, et optimiser leur rentabilité. Ce projet s'inscrit dans cette dynamique et vise à étudier l'évolution des ventes d'un produit sur une période donnée, à l'aide d'outils d'analyse chronologique.

L'objectif principal est de comprendre les tendances sous-jacentes, d'identifier les effets saisonniers ou cycliques, et de proposer un modèle prédictif capable d'anticiper les ventes futures. Cette démarche permettra non seulement de dégager des insights exploitables pour la prise de décision, mais également de démontrer la pertinence de l'approche statistique et des modèles de séries temporelles (tels que ARIMA et SARIMA) dans le domaine du marketing analytique.

À travers ce rapport, nous détaillerons les étapes méthodologiques suivies, de l'exploration des données à la modélisation, en passant par les transformations nécessaires, les tests de stationnarité, et l'évaluation des performances du modèle retenu.

La base de données a été trouvée sur le site Kaggle.

Elle contient les ventes de janvier 1964 vers septembre 1972.

Initialisation

Tout d'abord, pour commencer notre étude sur Python, j'ai commencé par l'importation de toutes les librairies statistiques dont j'aurais besoin.

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import datetime
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
✓ 0.0s
```

Matplotlib.pyplot + seaborn→ traçage des graphes

Pandas -→ pour travailler avec la base de données (l'importer, la nettoyer...)

Numpy → tout ce qui est mathématique

Adfuller → Test de Dickey-Fuller Augmenté

ARIMA, SARIMA

Plot_acf, plot_pacf → Traçage des fonctions d'autocorrélation et fonction autocorrélation partiel (resp)

A- Importation de la série et nettoyage :

```
data = pd.read_csv("sales.csv")
data.columns=["Date","Sales"]
data.asfreq("MS")
```

✓ 0.0s

```
data['Date'] = pd.to_datetime(data['Date'], errors='coerce')
# 1. S'assurer que les dates sont continues et sans doublons
data['Date'] = pd.to_datetime(data['Date'])
data = data.sort_values('Date').drop_duplicates('Date')
# 2. Mettre Date en index et forcer la fréquence trimestrielle
data.set_index('Date', inplace=True)
#3- Traitement des valeurs manquantes
first_valid = data.first_valid_index()      # repère la 1re date non-NaN
data = data.loc[first_valid:]          # coupe tout ce qui précède
data = data.interpolate(method = "linear", axis = 0, limit_direction= "both")
```

✓ 0.0s

J'importe tout d'abord la série « sales » qui est sous forme csv, en nommant ses colonnes : Date, Sales (ventes) et je précise la fréquence MS (début du mois), car les données sont précisées pour chaque début du mois.

Maintenant, il faut nettoyer la série. Tout d'abord je dis au logiciel que la colonne « date » est en « date time » et ensuite j'enlève les doublons

Pour traiter les données manquantes, j'ai décidé d'effectuer une interpolation linéaire.

B- Division de la série en : Train set et Test set :

```
split_index = int(len(data) * 0.90)
train_set = data.iloc[:split_index]
test_set = data.iloc[split_index:]
```
✓ 0.0s

J'ai décidé de travailler sur 90% de la série, laissant 10% pour la partie test.
Voici un aperçu sur les 2 séries :

Train set :

Date	Sales
1964-01-01	2815.0
1964-02-01	2672.0
1964-03-01	2755.0
1964-04-01	2721.0
1964-05-01	2946.0
...	...
1971-07-01	4633.0
1971-08-01	1659.0
1971-09-01	5951.0
1971-10-01	6981.0
1971-11-01	9851.0

95 rows × 1 columns

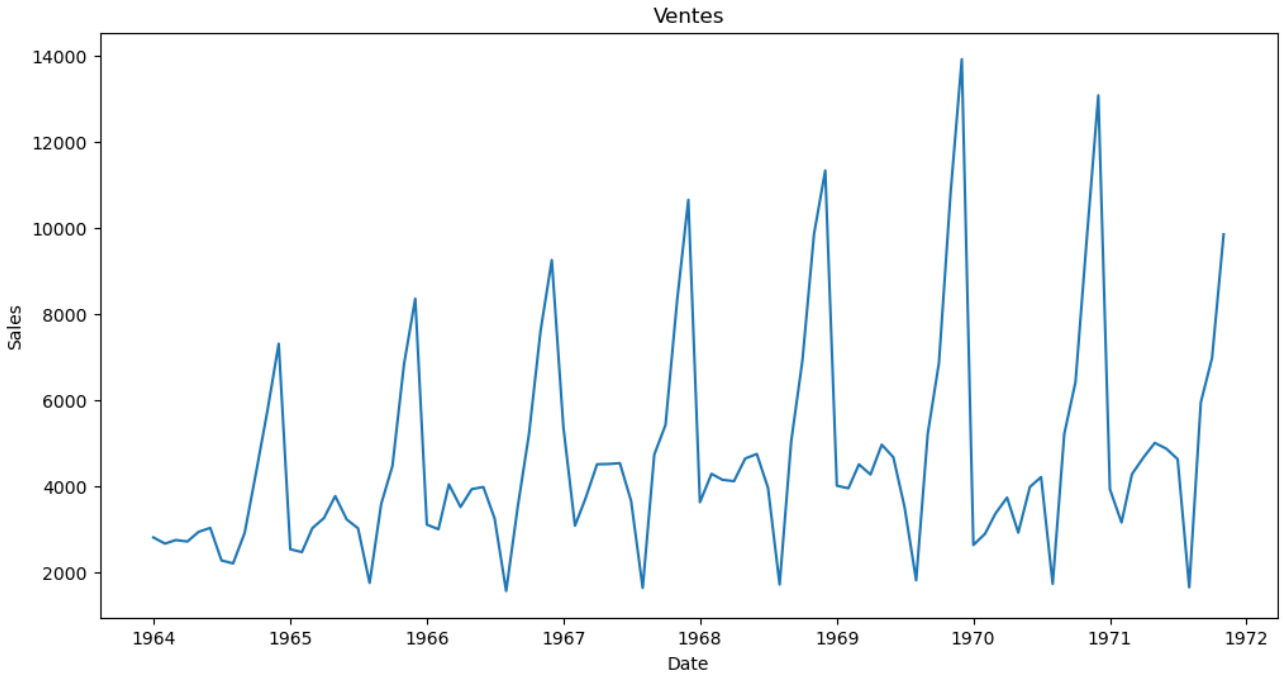
test set :

Date	Sales
1971-12-01	12670.0
1972-01-01	4348.0
1972-02-01	3564.0
1972-03-01	4577.0
1972-04-01	4788.0
1972-05-01	4618.0
1972-06-01	5312.0
1972-07-01	4298.0
1972-08-01	1413.0
1972-09-01	5877.0

Représentation graphique :

```
plt.figure(figsize=(12, 6))
sns.lineplot(data = train_set, x = 'Date', y= 'Sales')
plt.title("Ventes")
plt.show()
```

✓ 0.1s



A l'œil nu :

1 - ***Tendance générale :***

On observe une tendance haussière nette sur l'ensemble de la période.

Les valeurs minimales en 1964 sont autour de 2 000 ventes, tandis que les pics en 1970 dépassent 14 000 ventes.

Cela indique une croissance progressive et soutenue du produit étudié.

2- *Comportement saisonnier :*

Il existe des fluctuations récurrentes et régulières d'une année à l'autre.

Ces pics surviennent de manière périodique (environ tous les 12 mois), ce qui suggère une saisonnalité annuelle.

Par exemple, chaque année connaît un pic important suivi d'une chute brutale, indiquant peut-être un effet saisonnier de forte demande (ex : période de fêtes, rentrée, etc.).

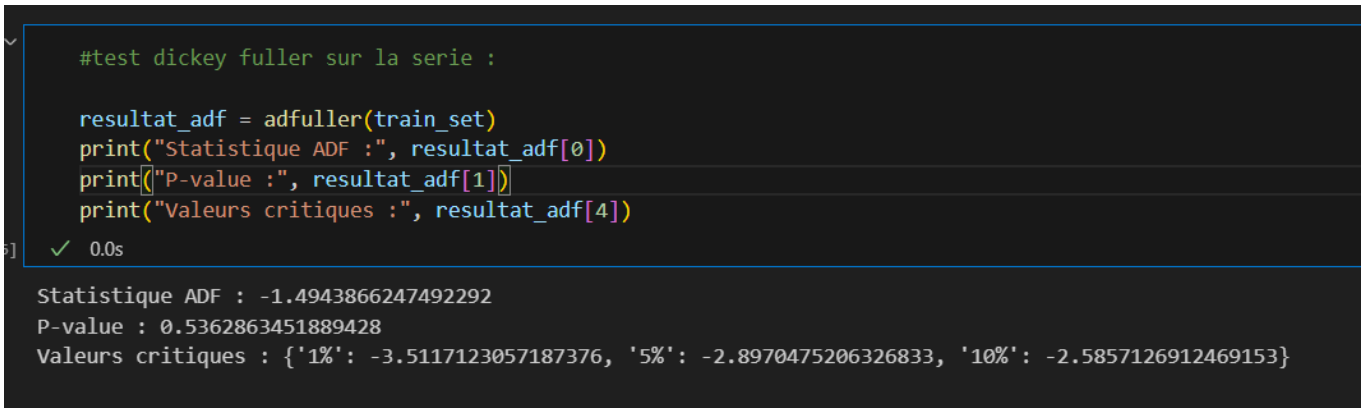
Test de stationnarité :

A – Test :

Maintenant qu'on a vu une possible tendance d'après la représentation graphique, il est temps de tester la stationnarité de la série, pour voir si effectivement elle n'est pas stationnaire.

Pour cela nous allons utiliser le test augmenté de Dickey-Fuller, qui test la stationnarité de la série, avec H0 : présence de racine unitaire.

Si on trouve P-value > 5%, cela veut dire que la série n'est pas stationnaire, si on trouve P-value <5% : Stationnarité validé



Ici, On trouve P-value = 0.5362 > 0.05 : La série n'est pas stationnaire.

Il faut maintenant **stationnariser la série**

B - Stationarisation de la série :

Pour stationnariser, on utilise la différenciation

$Y = (1-B^k)Xt$

Avec k l'ordre de différenciation (1,2,3,….)

On va maintenant différencier à l'ordre 1 t tester la stationnarité :

Sur PYTHON, on utilise la fonction .diff() :

```
#differenciation:
serie_diff = train_set.diff()
serie_diff = serie_diff.dropna()
```

✓ 0.0s

```
#test dickey fuller
resultat_adf = adfuller(serie_diff)
print("Statistique ADF :", resultat_adf[0])
print("P-value :", resultat_adf[1])
print("Valeurs critiques :", resultat_adf[4])
```

✓ 0.0s

```
Statistique ADF : -24.664513810945362
P-value : 0.0
Valeurs critiques : {'1%': -3.5117123057187376, '5%': -2.8970475206326833, '10%': -2.5857126912469153}
```

La P-value ici est : P-value = 0.000… < 0.05

Donc la série : serie_diff est bien stationnaire.

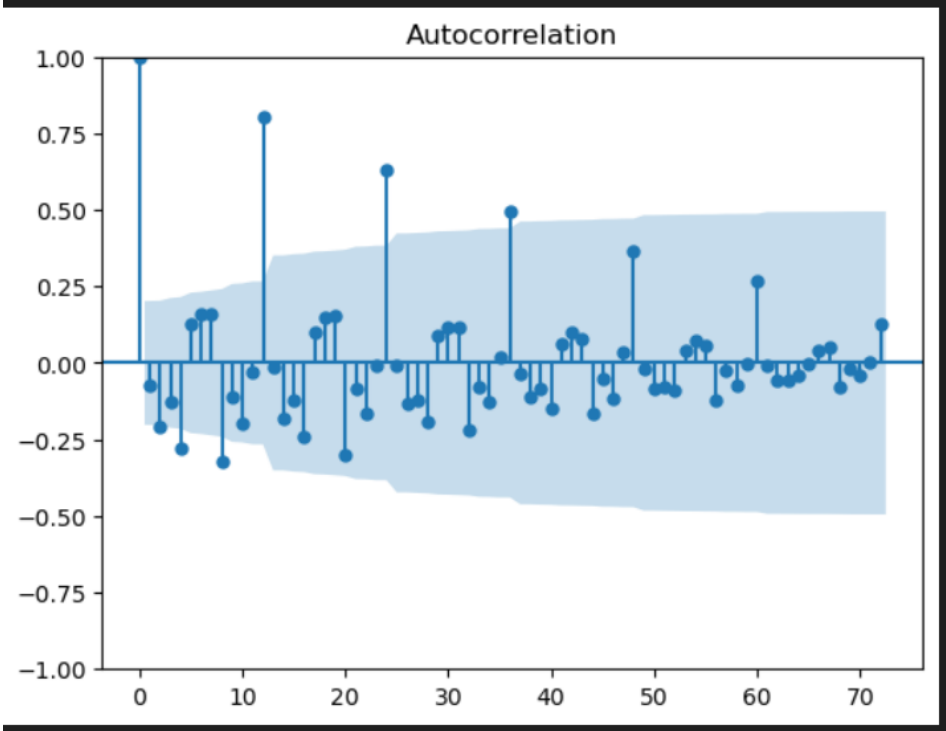
Ainsi on a réussi à stationnariser la série.

Le modèle devient donc un ARIMA(p,1,q) (d=1 : une seul différenciation)

FAC

On trace maintenant la fonction d'autocorrélation de la série : serie_diff qui est stationnaire, pour pouvoir en tirer des conclusions nécessaires pour la méthode ‘’Box et Jenkins’’

```
plot_acf(serie_diff, lags = 72)
plt.show()
```
✓ 0.0s



(La zone bleu s'agit d'un intervalle de confiance, on y reviendra lors de la méthode box et Jenkins)

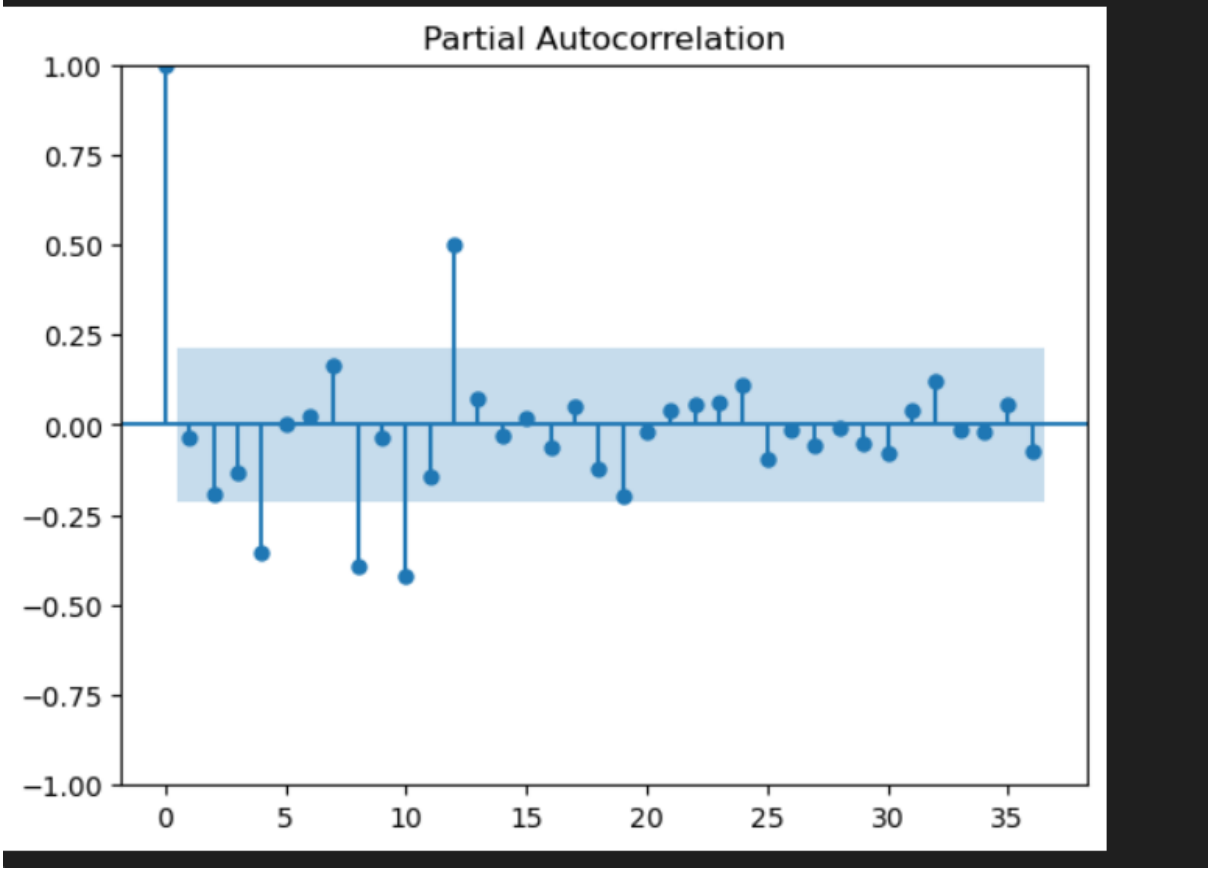
On remarque des pics chaque 12 mois, ce qui veut dire qu'**on a ici une saisonnalité de période 12.**

Ainsi, il faudrait travailler avec un SARIMA(p,1,q)(P,1,Q)

Or ici on remarque que ***les pics des périodes decroient de façon exponentielle***, ce qui veut dire que Q = 0 *Le premier pic est proche de 1 tandis que les suivants tendent vers 0, cela suggère une MA(1) (à vérifier)*

FACP

```
plot_pacf(serie_diff, lags = 36)
```
✓ 0.2s



Dans cette FACP, on remarque que le premier pic est proche de 1 et son voisin est presque significative, cela suggère une AR(1) (à vérifier)

On remarque ici que le pic 12 monte, mais les autres (24,36..) sont significatif, cela montre que P = 1

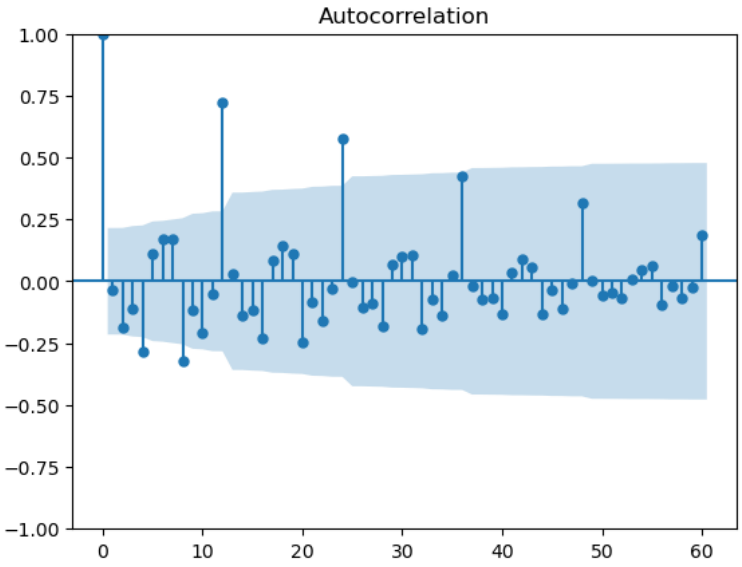
Ainsi, on travail avec un modèle SARIMA(p,1,q)(1,1,0) pour annuler la tendance et la saisonnalité.

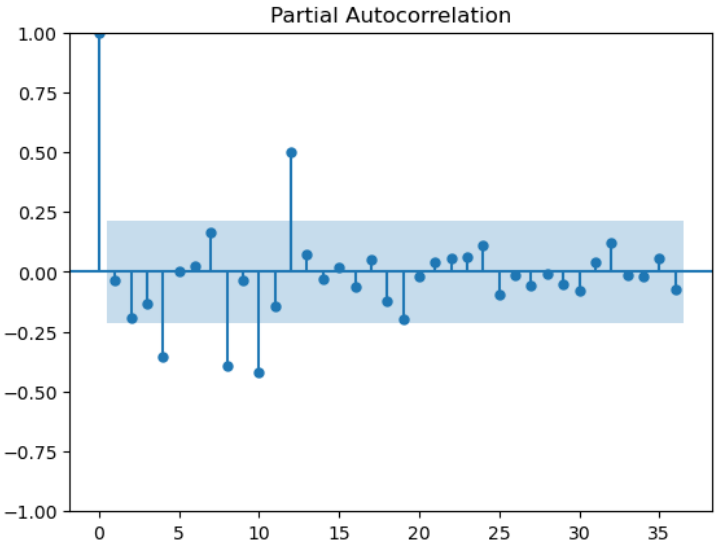
Il reste à déterminer p et q

Méthode de box-Jenkins :

Maintenant qu'on a notre SARIMA(p,1,q)(1,1,0) qui n'a ni tendance ni saisonnalité, il reste à identifier notre ARMA(p,q)

Revenons à la FAC et FACP





La zone bleue ici est l'intervalle de confiance pour le test de Bartlett, qui est :

$$\left[-\frac{z_{\alpha/2}}{\sqrt{T}}, \frac{z_{\alpha/2}}{\sqrt{T}}\right]$$

On choisit qmax premiers pics qui sont en dehors de l'intervalle de confiance dans la FAC, ici qmax étant qmax = 1

Et pmax premiers pics qui sont en dehors de l'intervalle de confiance dans la FACP, ici étant pmax = 1

On doit donc estimer les paramètres des modelés suivants :

SARIMA(1,1,1)(1,1,0,12), SARIMA(1,1,0)(1,1,0,12), SARIMA(0,1,1)(1,1,0,12)

Pour garder les modèles ou tout ses coefficients sont significatifs, et ensuite choisir celui qui a une AIC minimal, et le vérifier avec le test de LJUNG-BOX

Estimons les modèles avec la fonction suivante :

```
model = SARIMAX(train_set, order=(1,1,1), seasonal_order = (1,1,0,12))
res   = model.fit()
print(res.summary())
```

Tout en changeant l'ordre (1,1,1) par les autres.

L'estimation pour SARIMA(1,1,1)(1,1,0,12) donne :

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                           Sales   No. Observations:                   95
Model:             SARIMAX(1, 1, 1)x(1, 1, [], 12)   Log Likelihood                -661.731
Date:                             Sat, 24 May 2025   AIC                           1331.462
Time:                                     19:44:01   BIC                           1341.089
Sample:                                 01-01-1964   HQIC                          1335.327
                                      - 11-01-1971
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2932      0.077      3.817      0.000       0.143       0.444
ma.L1         -0.9432      0.051    -18.434      0.000      -1.043      -0.843
ar.S.L12      -0.2529      0.092     -2.758      0.006      -0.433      -0.073
sigma2      5.465e+05   6.87e+04      7.952      0.000    4.12e+05    6.81e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.26   Jarque-Bera (JB):                 7.51
Prob(Q):                              0.61   Prob(JB):                         0.02
Heteroskedasticity (H):               2.01   Skew:                            -0.27
Prob(H) (two-sided):                  0.07   Kurtosis:                         4.39
===================================================================================
```

Notons que AIC = 1331.462, les coefficients AR et MA et sigma2 vérifient le test de significativité de student (tstudent > 1.96) (étant z sur l'image)

Aussi, le test de LJUNG-BOX est vérifié car P-value > 0.05

On a ici un modèle candidat.

Les autres modèles :

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                             Sales   No. Observations:                   95
Model:             SARIMAX(1, 1, 0)x(1, 1, 0, 12)   Log Likelihood                -671.905
Date:                            Sat, 24 May 2025   AIC                           1349.810
Time:                                    19:49:45   BIC                           1357.030
Sample:                                01-01-1964   HQIC                          1352.708
                                     - 11-01-1971
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.3107      0.078     -3.991      0.000      -0.463      -0.158
ar.S.L12      -0.2682      0.075     -3.569      0.000      -0.415      -0.121
sigma2      7.281e+05   8.01e+04      9.091      0.000    5.71e+05    8.85e+05
===================================================================================
Ljung-Box (L1) (Q):                   2.04   Jarque-Bera (JB):                38.31
Prob(Q):                              0.15   Prob(JB):                         0.00
Heteroskedasticity (H):               1.79   Skew:                            -1.00
Prob(H) (two-sided):                  0.14   Kurtosis:                         5.69
===================================================================================
```

SARIMA(0,1,1)(1,1,0,12) vérifie aussi toutes les conditions, et son AIC =1349.810, qui est plus grande que l'ancien modèle

Donc SARIMA(1,1,1)(1,1,0,12) reste le candidat

Étudions SARIMA(1,1,0)(1,1,0,12) :

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                             Sales   No. Observations:                   95
Model:             SARIMAX(0, 1, 1)x(1, 1, [], 12)   Log Likelihood                -664.982
Date:                            Sat, 24 May 2025   AIC                           1335.963
Time:                                    19:51:06   BIC                           1343.183
Sample:                                01-01-1964   HQIC                          1338.862
                                     - 11-01-1971
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -0.8604      0.056    -15.415      0.000      -0.970      -0.751
ar.S.L12      -0.2033      0.088     -2.301      0.021      -0.376      -0.030
sigma2      5.816e+05   7.4e+04      7.861      0.000    4.37e+05    7.27e+05
===================================================================================
Ljung-Box (L1) (Q):                   2.40   Jarque-Bera (JB):                 2.57
Prob(Q):                              0.12   Prob(JB):                         0.28
Heteroskedasticity (H):               2.01   Skew:                             0.06
Prob(H) (two-sided):                  0.08   Kurtosis:                         3.86
===================================================================================
```

Il vérifie toutes les conditions

AIC = 1335.963 > 1331.42

Ainsi, le modèle final est SARIMA(1,1,1)(1,1,0,12)

Prédiction :

Maintenant qu'on a notre modèle, on va l'utiliser pour faire des prévisions, et il est temps d'utiliser notre dataset : test set

Je vais faire des prévisions pour la même période présente dans le test set, et le but est de comparer la prévision a la réalité, et ainsi voir la précision du modèle.

On va aussi calculer l'erreur quadratique moyenne et le mean absolute déviation.

```
h = len(test_set)
fcst_obj = model.get_forecast(steps=h)
pred     = fcst_obj.predicted_mean
IC = fcst_obj.conf_int()              # IC 95 %

# ---------------- Métriques RMSE / MAE ----------------
from sklearn.metrics import mean_squared_error, mean_absolute_error
rmse = mean_squared_error(test_set, pred, squared=False)
mae  = mean_absolute_error(test_set, pred)
print(f"RMSE : {rmse:,.2f}   |   MAE : {mae:,.2f}")
```

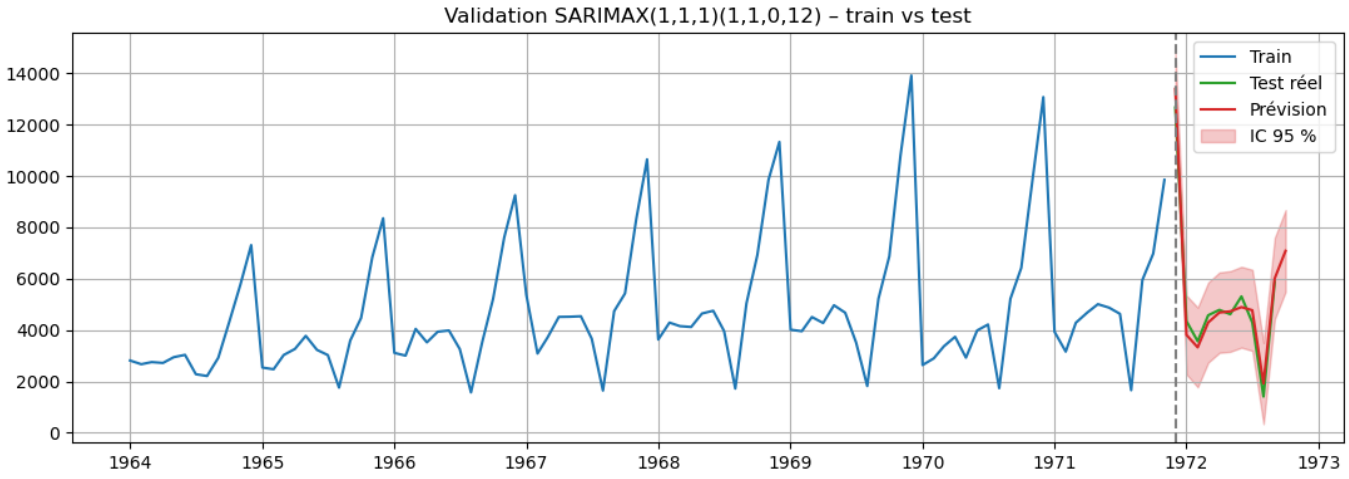
✓ 0.0s

```
RMSE : 531.74   |   MAE : 430.79
```

La fonction get_forecast(steps =h) défini la prédiction du modèle, avec model étant le SARIMA(1,1,1)(1,1,0,12), et h étant l'horizon qui est égal à la longueur de la base de test.

On trouve RMSE = 531.74 et MAE=430.79

Représentation graphique : prédiction vs réalité

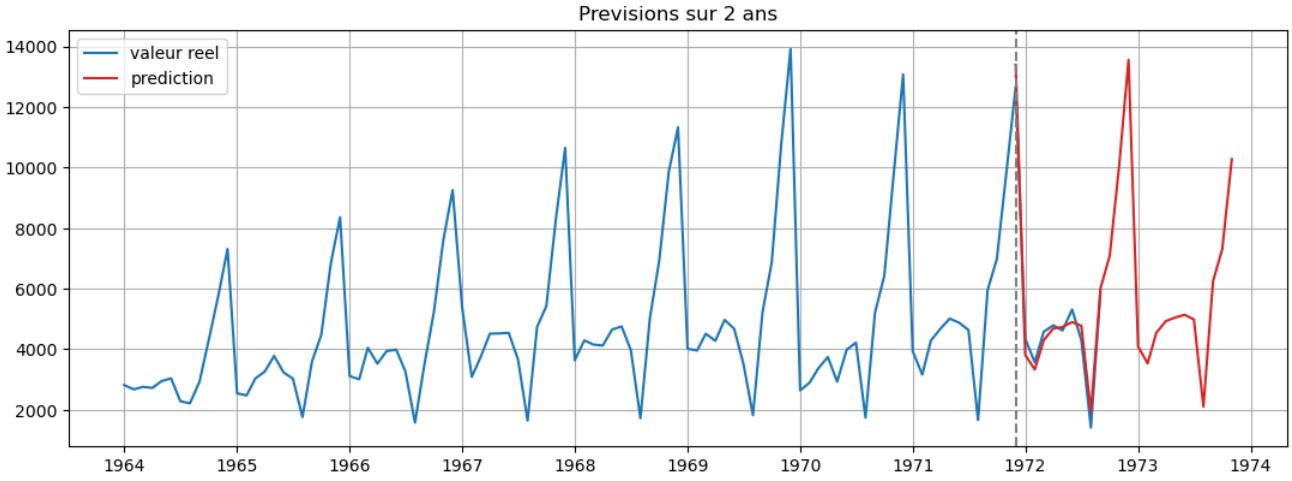


Le graphique ci-dessus illustre la performance du modèle SARIMA(1,1,1)(1,1,0,12) en comparant les valeurs réelles de la série de test avec les valeurs prédites, accompagnées de leur intervalle de confiance à 95 %.

Observations :

- Le modèle parvient à capturer correctement la tendance et la saisonnalité de la série.
- Les pics de ventes annuels sont bien anticipés.
- Les prévisions restent globalement dans l'intervalle de confiance à 95 %, ce qui témoigne d'un bon ajustement global.

Prévision sur 2 ans :



En 2ans on prédit une vente minimale de 2000 articles et une vente maximal d'environ 13500 articles.

Ces données sont essentielles pour la société de production car elles constituent la base de la planification opérationnelle et financière : comment gérer le stock, combien produire pour chaque période pour ne pas avoir de pertes, planification des ressources humaines…

CONCLUSION :

Ce travail avait pour objectif d'appliquer la méthode de Box et Jenkins afin de modéliser et prévoir l'évolution des ventes d'un produit à partir d'une série chronologique réelle. L'ensemble du processus d'analyse a été rigoureusement mené, depuis la visualisation initiale de la série jusqu'à la validation des prévisions obtenues.

L'analyse exploratoire a permis de mettre en évidence une **tendance haussière claire** ainsi qu'une **saisonnalité annuelle marquée**, justifiant le recours à un modèle de type **SARIMA**. Après avoir rendu la série stationnaire par différenciation, les corrélogrammes (ACF et PACF) ont guidé le choix des ordres du modèle.

Le modèle SARIMA(1,1,1)(1,1,0,12) s'est révélé particulièrement performant pour capturer la dynamique de la série. Les prévisions réalisées sur l'échantillon de test ont montré une bonne concordance avec les données réelles, avec des **erreurs de prévision acceptables** :

- **RMSE = 531.74**
- **MAE = 430.79**

Ces résultats démontrent la **pertinence du modèle choisi** et valident la méthodologie utilisée. Le modèle peut ainsi être utilisé pour effectuer des prévisions fiables à court terme, et servir d'outil d'aide à la décision pour la gestion de la production, des stocks ou des campagnes marketing.