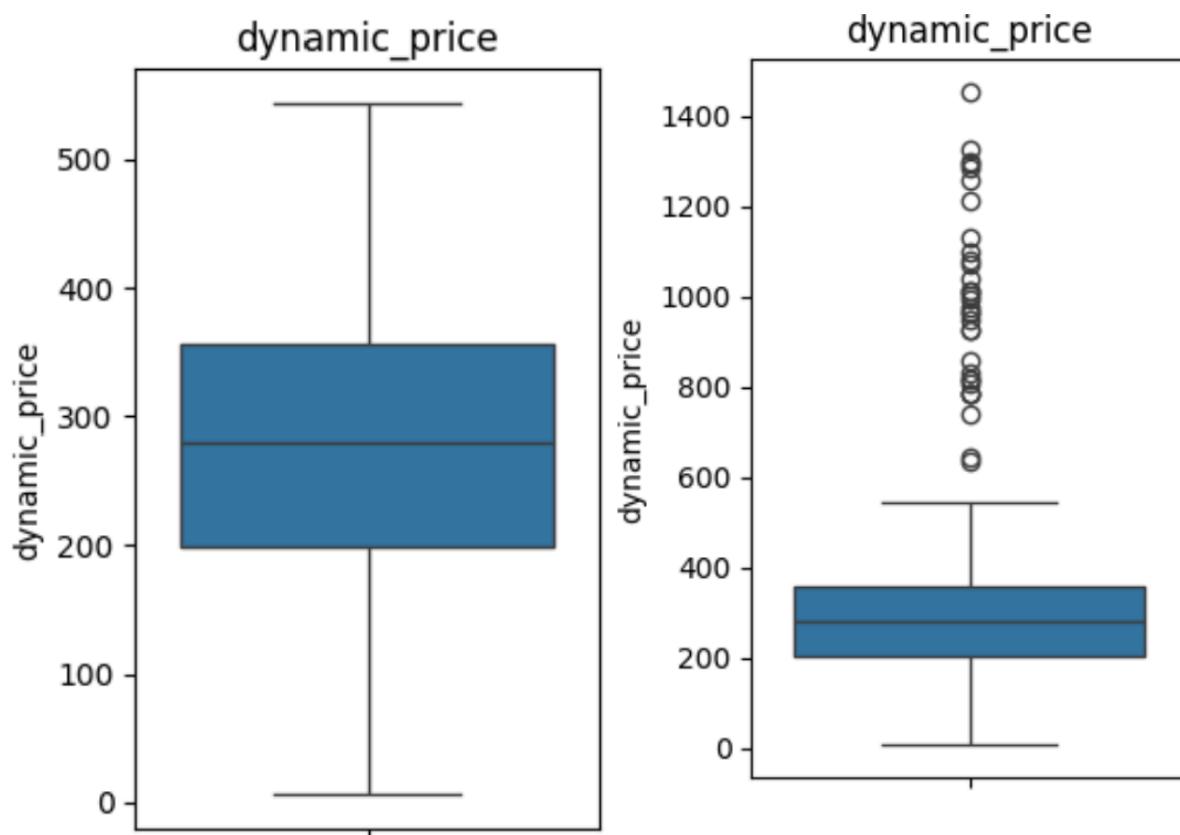


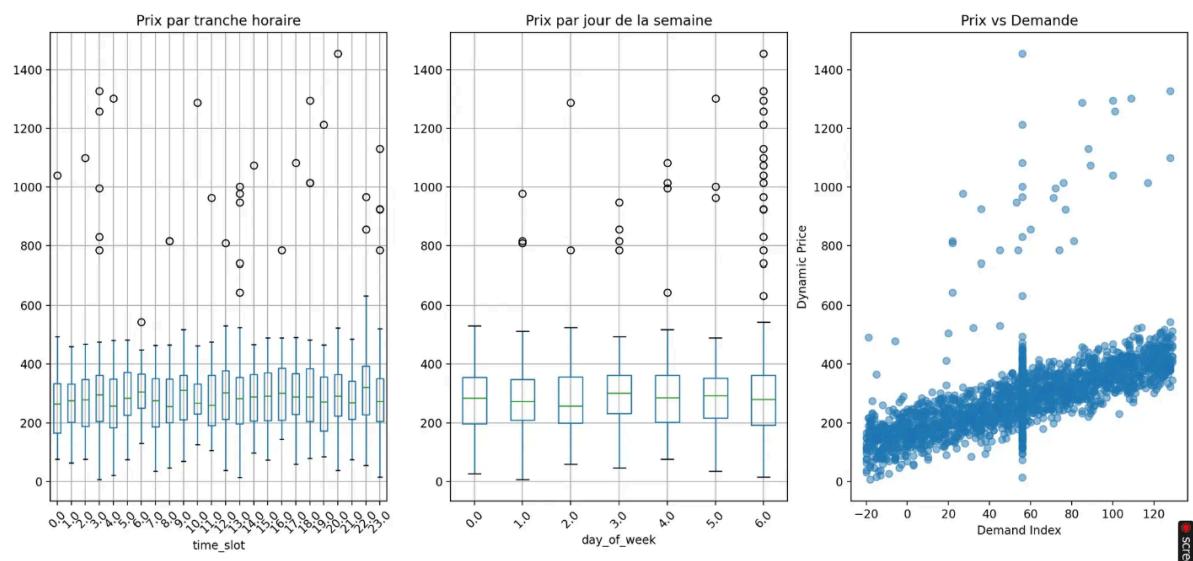
# Rapport de visualisation et interface

## Analyse de la variable cible



- **Sans outliers** : distribution plus homogène, médiane plus stable.
- **Avec outliers** : nombreuses valeurs extrêmes (~600 à 1400) qui biaisent les analyses statistiques et le modèle ML.
- **Conclusion** : le nettoyage améliore la lisibilité et la fiabilité des analyses (1)

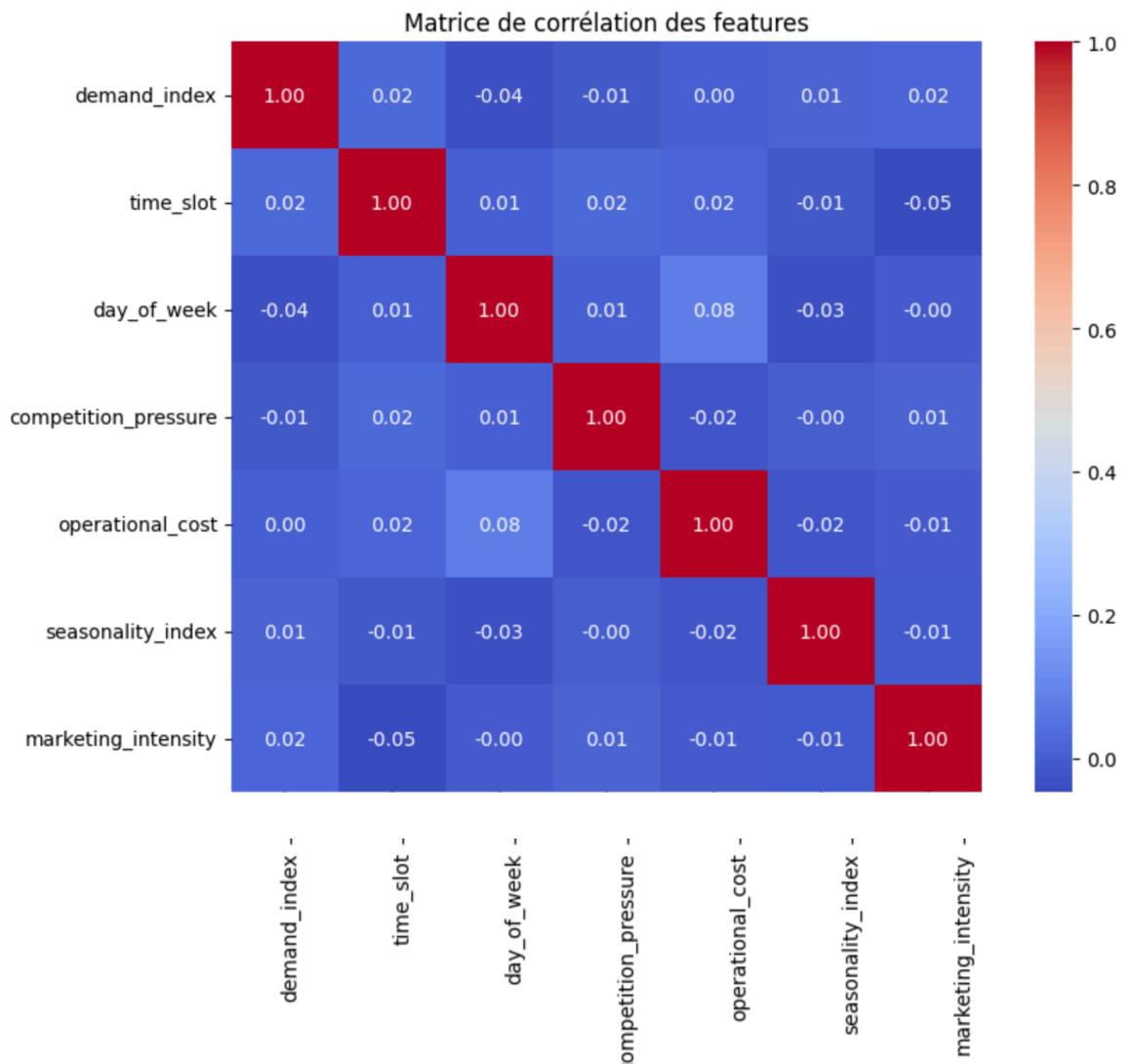
## Analyse des Outliers du variable cible par Feature



Les visualisations montrent que les outliers de la variable cible sont répartis sur **différentes valeurs des features** (heure, jour, demande).

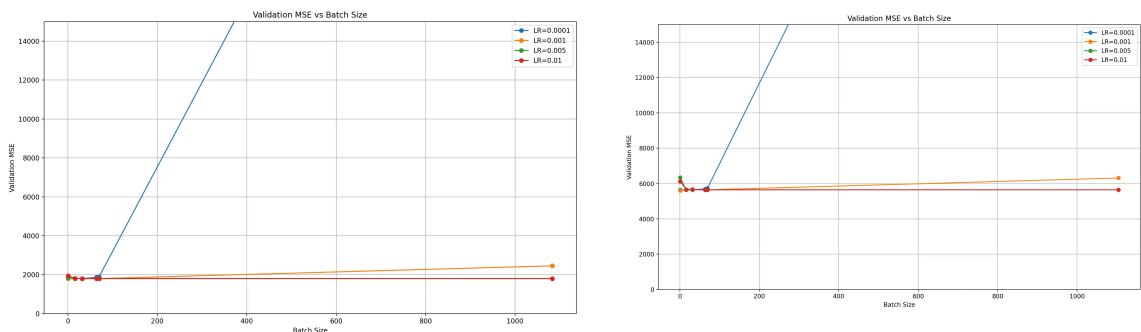
Cela indique qu'il **n'existe pas de lien direct ou exclusif** entre les outliers et une modalité particulière.

## Corrélation entre variables et réduction de dimension



Les variables étant faiblement corrélées (valeur absolue de corr entre 0 et 0.08), l'ACP n'est pas adaptée. à ce contexte.

## Analyse des hyperparamètres et impact des outliers



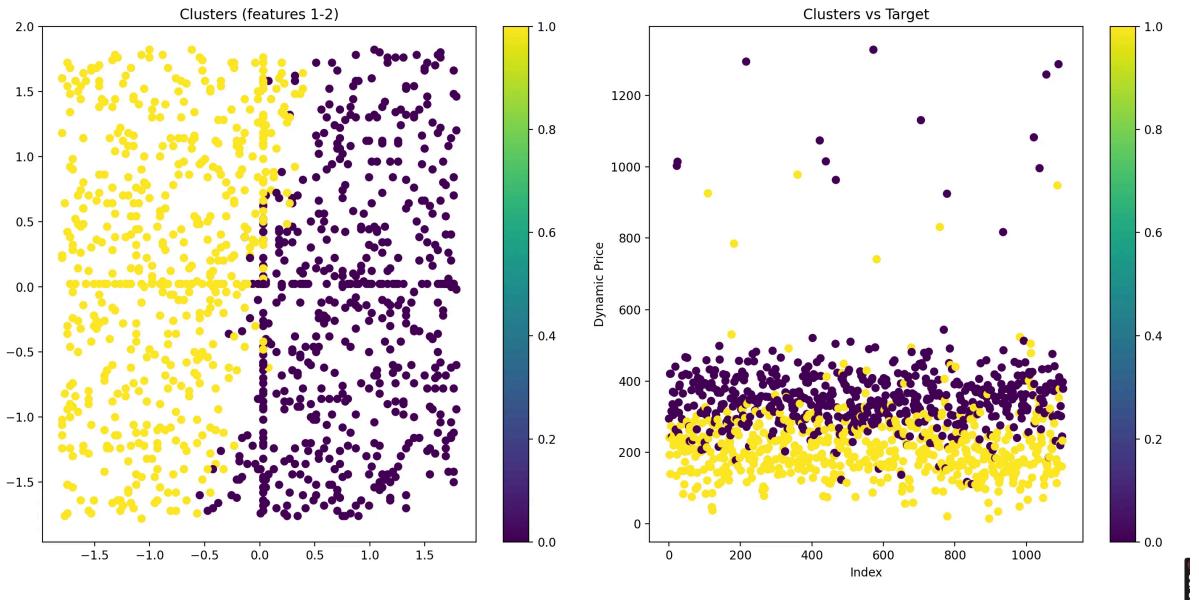
Ce graphique est utile pour choisir des hyperparamètres stables. Il montre que :

- Un petit learning rate avec un grand batch size peut ralentir l'apprentissage et conduire à une **divergence** du MSE.
- Une taille de batch modérée (ni trop petite ni trop grande) combinée à un taux d'apprentissage bien calibré (autour de 0.001 à 0.005) offre un bon équilibre.
- **Sans outliers** : MSE optimal  $\approx 1700$
- **Avec outliers** : MSE optimal  $\approx 5600$
- Les deux graphiques (avec et sans outliers) présentent une allure de variation similaire ; la différence observée correspond principalement à une **translation** verticale sur l'axe du MSE, due à l'augmentation de la variance induite par les valeurs aberrantes.

### **Conclusion :**

Le choix de la méthode d'optimisation (BGD, Mini-batch GD ou SGD) des paramètres du modèle (poids et biais) dépend davantage du réglage des hyperparamètres, en particulier du learning rate, que des données d'entraînement elles-mêmes. Ces hyperparamètres influencent directement la qualité de convergence et la valeur finale du MSE.

## Clustering-based regression



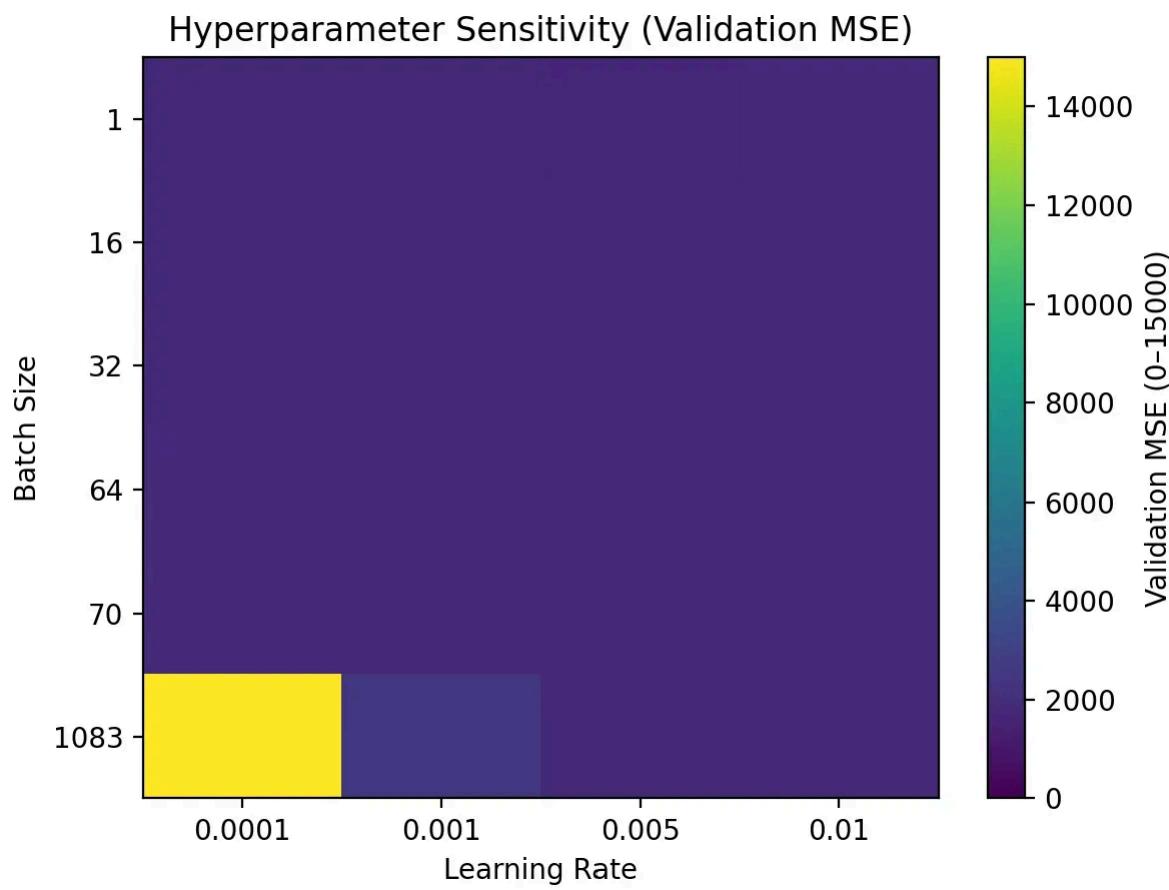
- **Silhouette score 0.1 (K-means)** : Clustering faible, clusters chevauchants et mal séparés.
- **Graphe gauche** : Clusters jaune/violet fortement superposés en 2D (faible séparation features).
- **Graphe droit** : Clusters mélangés vs target (violet tend hautes valeurs, mais pas de nettes bandes)

### Conclusion :

Le clustering K-means (silhouette 0.1) échoue à segmenter efficacement les données en raison de clusters fortement chevauchants causés par les outliers, rendant l'approche clustering-then-regression peu bénéfique pour améliorer la performance de régression.

→ Prioriser la suppression des outliers plutôt que leur gestion.

## Comparaison des méthodes de Gradient Descent



### SGD (Stochastic Gradient Descent)

Donne de **bons MSE uniquement quand le learning rate est petit**

SGD met à jour les poids **à chaque observation**(gradient très bruité).

Avec un **grand LR**, les mises à jour sont trop violentes → oscillations → MSE élevé.

→ SGD fonctionne bien avec un learning rate faible

### Mini-Batch Gradient Descent

Donne de **bons MSE pour presque tous les learning rates**

- réduit le bruit du gradient (vs SGD)
- garde une bonne fréquence de mise à jour (vs BGD)

→ Mini-Batch GD est robuste et peu sensible au learning rate

## BGD (Batch Gradient Descent)

Donne de **bons résultats quand le learning rate est grand**

BGD fait **une seule mise à jour par époque**, avec un gradient très précis.

- LR petit → pas trop faible → convergence **très lente**
- LR grand → permet d'avancer efficacement

→ BGD fonctionne bien avec un learning rate élevé

## Fine tunning des hyperparamètres

```
pc@LAPTOP-MQ7RFLOI MINGW64 ~/Desktop/Bootcamp/projet/ml-project/part1_dynamic_pricing
$ python train_model_without_target_outliers.py

Hyperparameter tuning with validation set...

LR=0.0001 | SGD           | Val MSE=1801.2978 | Val R2=0.8084
LR=0.0001 | Mini-Batch (16) | Val MSE=1801.0170 | Val R2=0.8085
LR=0.0001 | Mini-Batch (32) | Val MSE=1799.0364 | Val R2=0.8087
LR=0.0001 | Mini-Batch (64) | Val MSE=1864.6113 | Val R2=0.8017
LR=0.0001 | Mini-Batch (70) | Val MSE=1890.6655 | Val R2=0.7989
LR=0.0001 | Batch GD        | Val MSE=45818.4246 | Val R2=-3.8727
LR=0.001   | SGD             | Val MSE=1798.3599 | Val R2=0.8087
LR=0.001   | Mini-Batch (16) | Val MSE=1801.0632 | Val R2=0.8085
LR=0.001   | Mini-Batch (32) | Val MSE=1801.1267 | Val R2=0.8085
LR=0.001   | Mini-Batch (64) | Val MSE=1801.0224 | Val R2=0.8085
LR=0.001   | Mini-Batch (70) | Val MSE=1801.2353 | Val R2=0.8084
LR=0.001   | Batch GD        | Val MSE=2453.4536 | Val R2=0.7391
LR=0.005   | SGD             | Val MSE=1757.0825 | Val R2=0.8131
LR=0.005   | Mini-Batch (16) | Val MSE=1801.8262 | Val R2=0.8084
LR=0.005   | Mini-Batch (32) | Val MSE=1801.8309 | Val R2=0.8084
LR=0.005   | Mini-Batch (64) | Val MSE=1801.2528 | Val R2=0.8084
LR=0.005   | Mini-Batch (70) | Val MSE=1802.1353 | Val R2=0.8083
LR=0.005   | Batch GD        | Val MSE=1800.8504 | Val R2=0.8085
LR=0.01    | SGD             | Val MSE=1853.5228 | Val R2=0.8029
LR=0.01    | Mini-Batch (16) | Val MSE=1798.5092 | Val R2=0.8087
LR=0.01    | Mini-Batch (32) | Val MSE=1800.5688 | Val R2=0.8085
LR=0.01    | Mini-Batch (64) | Val MSE=1801.2687 | Val R2=0.8084
LR=0.01    | Mini-Batch (70) | Val MSE=1801.9423 | Val R2=0.8084
LR=0.01    | Batch GD        | Val MSE=1801.1655 | Val R2=0.8084

Final Test Performance
Test MSE: 1848.1269
Test R2 : 0.8222

Best Model Found (Validation)
learning_rate: 0.005
batch_size: 1
method: SGD
val_r2: 0.8131377460594313
Best Val MSE: 1757.0825
(base)
pc@LAPTOP-MQ7RFLOI MINGW64 ~/Desktop/Bootcamp/projet/ml-project/part1_dynamic_pricing
$ |
```

le fine-tuning des hyperparamètres a permis d'identifier le modèle optimal, correspondant à un learning rate de 0.005 et à une stratégie sgd (batch size = 1).

ce modèle obtient les meilleures performances sur l'ensemble de validation, avec un **r<sup>2</sup> = 0.81** et un **mse = 1757**.

lors de l'évaluation sur l'ensemble de test, le modèle conserve de bonnes capacités de généralisation, avec un **mse = 1848** et un **r<sup>2</sup> = 0.82**, ce qui confirme la stabilité du réglage des hyperparamètres et l'absence de surapprentissage significatif.

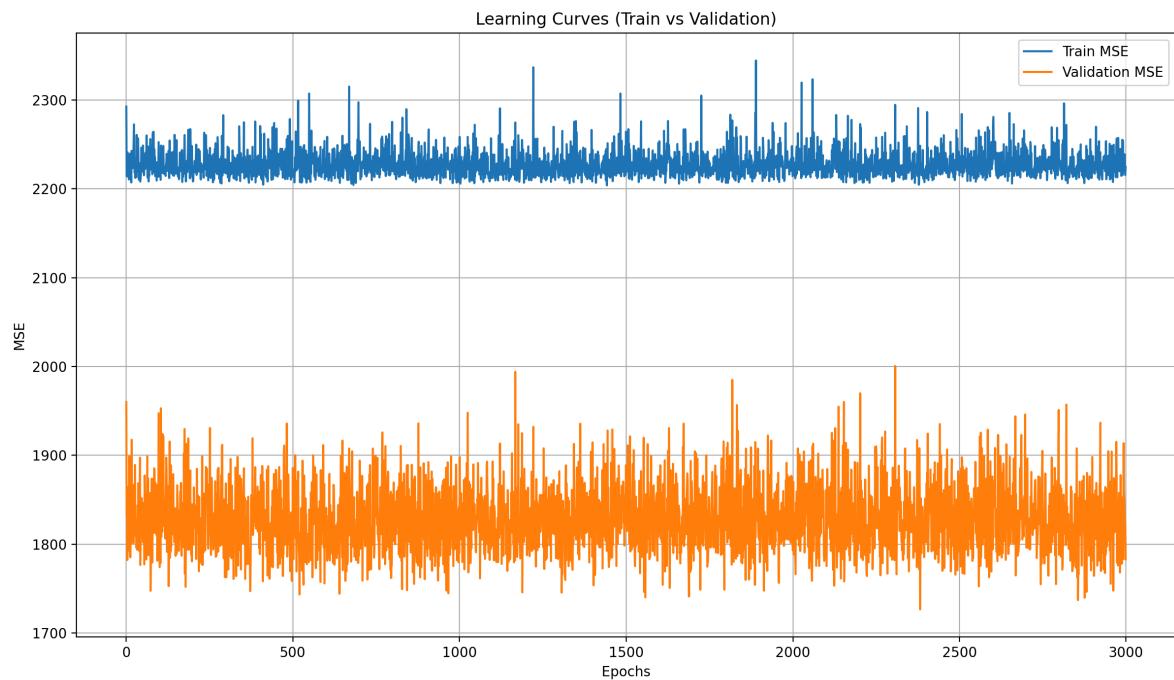
- ratio de précision : ratio = rmse / sd  $\approx 2.4$
- part de variance expliquée :  $R^2 \approx 1 - (sd^2 / rmse^2) \approx 0.83$

→ le modèle explique **environ 82–83 % de la variance**, ce qui est **cohérent avec r<sup>2</sup> ≈ 0.82**



le rmse obtenu sur l'ensemble de test ( $\approx 42$ ) est nettement inférieur à l'écart-type de la variable cible ( $\approx 101$ ), soit une erreur environ **2,4 fois plus faible que la variabilité intrinsèque des données**. cette relation correspond à une **variance expliquée d'environ 82 %**, confirmant que le modèle capture efficacement la structure des données et fournit des prédictions significativement plus précises qu'un modèle naïf basé sur la moyenne.

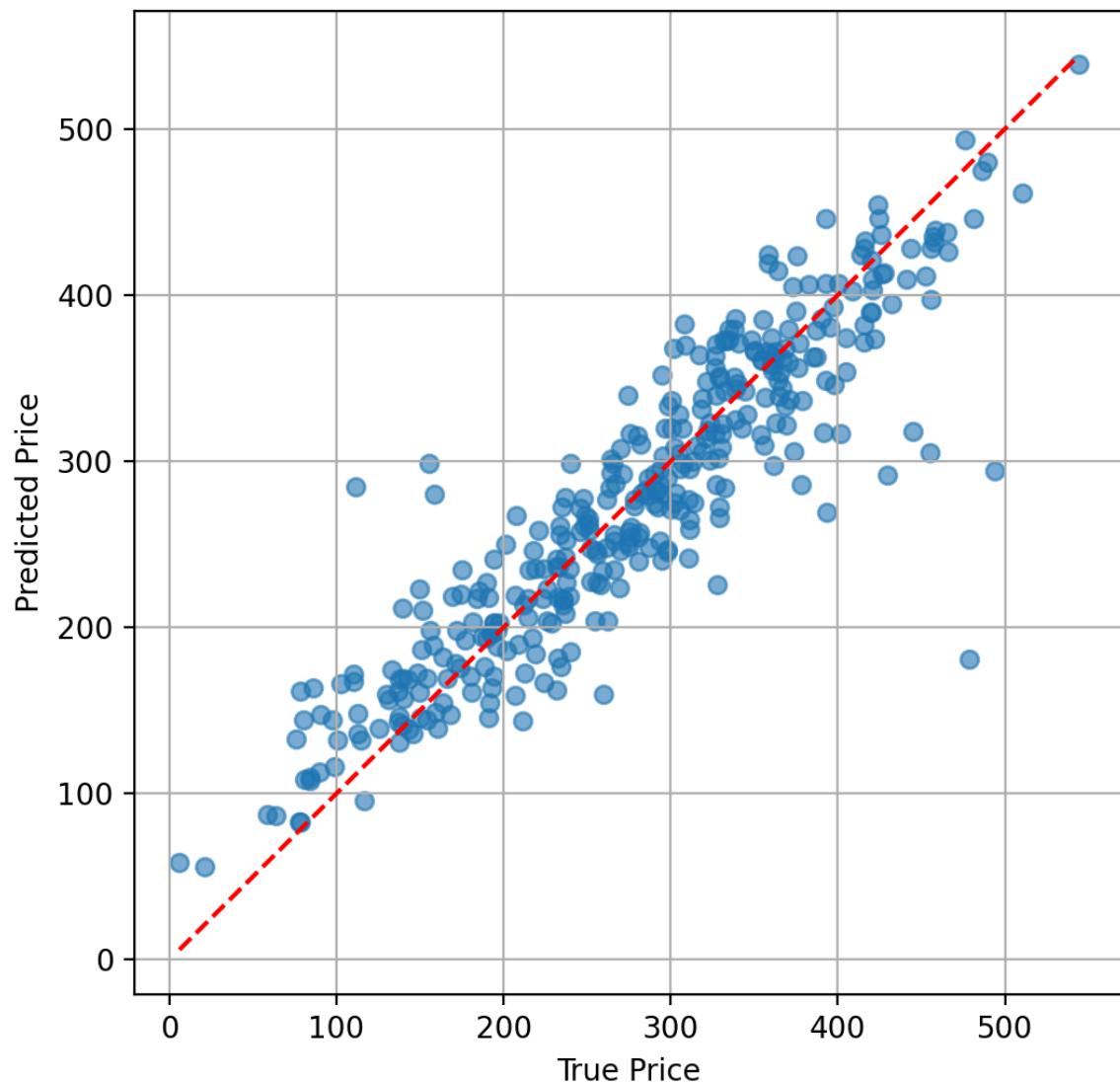
## Learning Curves

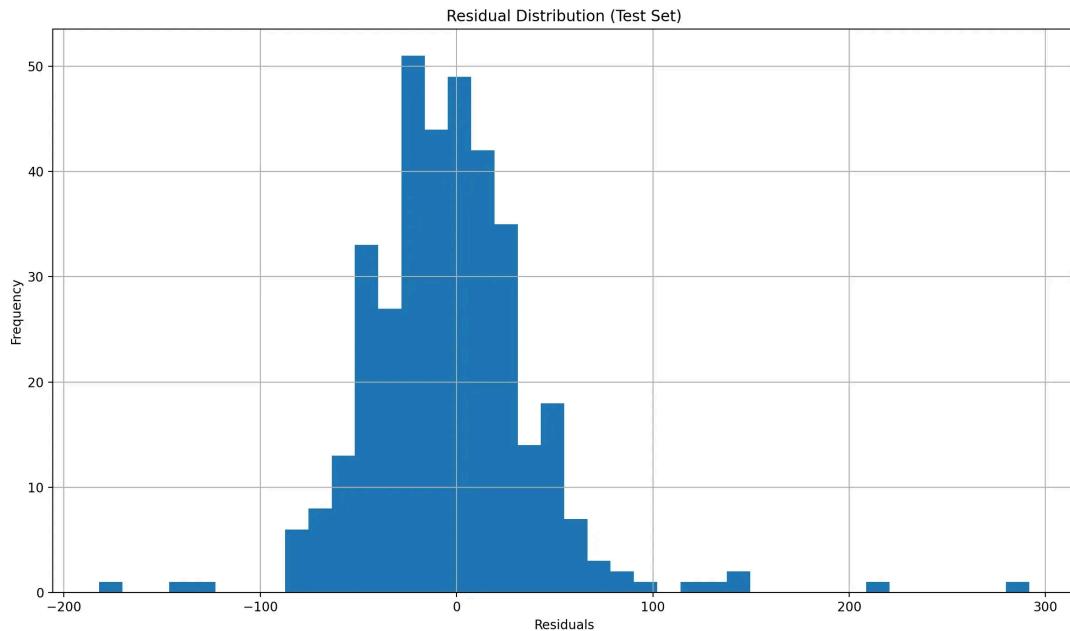


Les learning curves montrent que le modèle généralise bien, sans signe de surapprentissage.

## Évaluation du modèle sur le jeu de test

Predicted vs True Prices (Test Set)





- La majorité des résidus est **concentrée autour de 0**
- La distribution est **quasi gaussienne** au centre
- On observe une **queue de longueur moyenne à droite aussi à gauche**

### 👉 Interprétation

- le modèle **n'est pas biaisé globalement**
- les prédictions sont, en moyenne, **correctes**
- il ne capture pas bien les **comportements extrêmes avec un biais au maximum de 300.**
- Le modèle prédit parfois des valeurs sous-estimées et parfois surestimées. On peut donc dire qu'il est **non conservatif**, car ses erreurs ne vont pas toujours dans le même sens.

## Frontend et exemple d'utilisation (test en local)

# Prediction de Prix Dynamiques

Modele de regression par descente de gradient

Test R2

**0.8255**

Test MSE

**1813.91**

Indice de Demande

53.0

Intensite de la demande client

Pression Concurrentielle

1.55

Niveau de concurrence sur le marche

Cout Operationnel

110.01

Couts d'exploitation

Indice de Saisonnalite

0.62

Impact de la saison

Intensite Marketing

Niveau d'effort marketing

Creneau Horaire

13:00 - 14:00

Jour de la Semaine

Dimanche

**Predire le Prix**

**198.45 MAD**

<https://github.com/fatima-ezzahra-kadiri/dynamic-pricing-app>

## Le déploiement sous Render

Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. [Upgrade now](#)

December 30, 2025 at 11:16 PM [Live](#)  
f3f6133 Fix: update dependencies

All logs  [Live tail](#) [GMT+1](#) [e](#) ...

```

Dec 30
11:19:36 PM [rxc69] =====
11:19:36 PM [rxc69] Test MSE: 1813.9098
11:19:36 PM [rxc69] Test R2: 0.8255
11:19:36 PM [rxc69] Learning Rate: 0.005
11:19:36 PM [rxc69] Batch Size: 1
11:19:36 PM [rxc69] =====
11:19:36 PM [rxc69] 127.0.0.1 - - [30/Dec/2025:22:19:36 +0000] "HEAD / HTTP/1.1" 200 0 "-" "Go-http-client/1.1"
11:19:40 PM => Your service is live 🚀
11:19:40 PM =>
11:19:40 PM => ///////////////////////////////////////////////////
11:19:40 PM =>
11:19:40 PM => Available at your primary URL https://dynamic-pricing-app-tkhs.onrender.com
11:19:40 PM =>
11:19:40 PM => ///////////////////////////////////////////////////
11:19:42 PM [rxc69] 127.0.0.1 - - [30/Dec/2025:22:19:42 +0000] "GET / HTTP/1.1" 200 13682 "-" "Go-http-client/2.0"

```

Need better ways to work with logs? Try the [Render CLI](#), [Render MCP Server](#), or set up a [log stream integration](#).

Enfin, notre système est bien accessible sous l'URL :

<https://dynamic-pricing-app-tkhs.onrender.com/>

## Prediction de Prix Dynamiques

Modele de regression par descente de gradient

Test R2	Test MSE
<b>0.8255</b>	<b>1813.91</b>

Indice de Demande	Pression Concurrentielle	Cout Operationnel
54	1.5	110

Intensite de la demande client Niveau de concurrence sur le marché Couts d'exploitation

Indice de Saisonnalité	Intensite Marketing	Creneau Horaire	Jour de la Semaine
0.60		13:00 - 14:00	Dimanche

Impact de la saison Niveau d'effort marketing

**Predire le Prix**

**200.88 MAD**