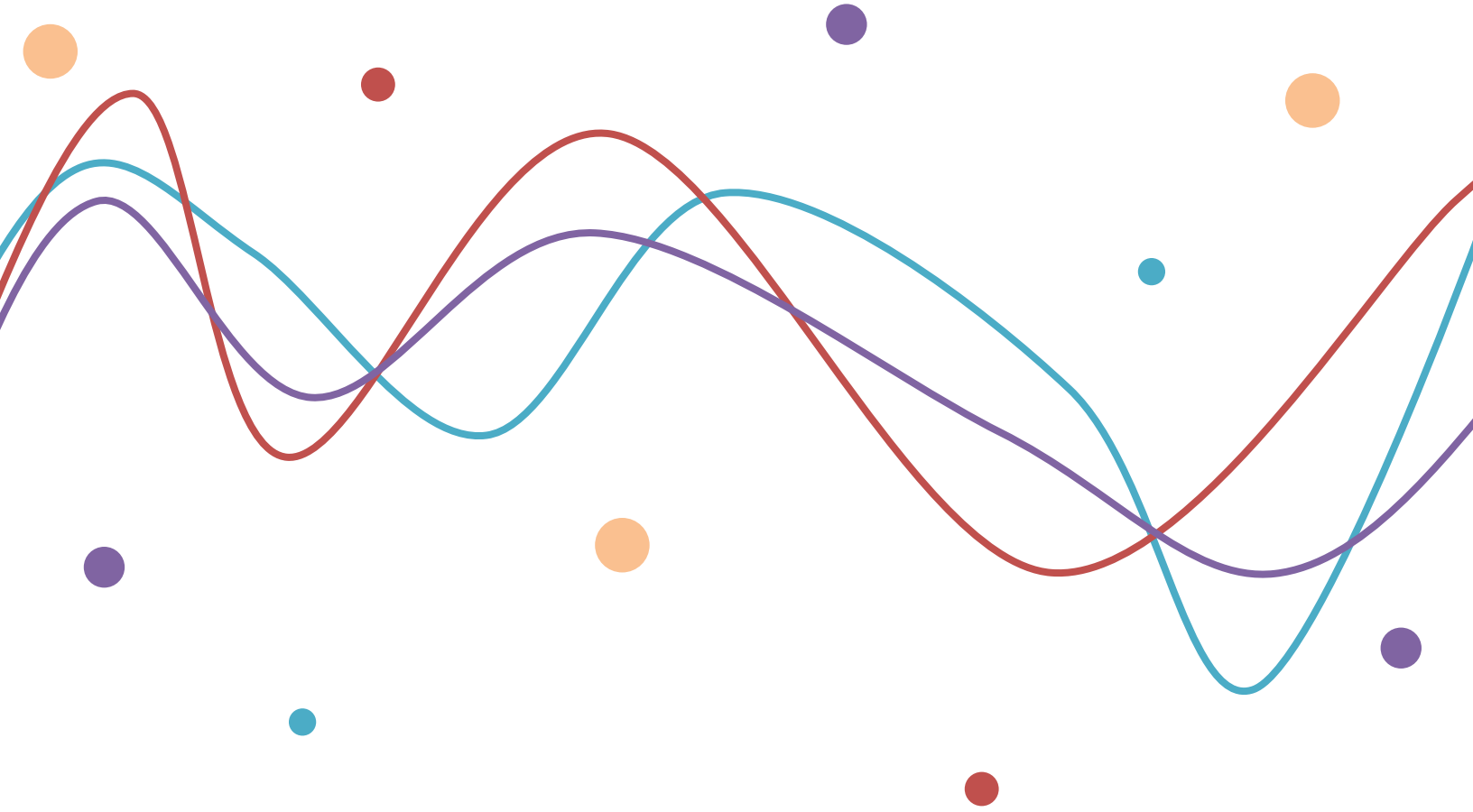
Rapport de projet
Simulation de Monte Carlo



Professeur : M. Hassan MAATOUK

Formation : ING1 GMA

Étudiants : Charles Meldhine MADI MNEMOI, Youssef SAIDI, Lucas TERRA et Marwa TOURABI

Sommaire

1. Choix des technologies

N'ayant pas de restrictions au niveau de la sélection des outils pour l'application de la Simulation de Monte Carlo, nous avons décidé d'utiliser le langage de programmation Python.

Python est un langage de programmation extrêmement utile et polyvalent qui est couramment utilisé pour l'analyse, le traitement et la visualisation des données, mais également dans le développement de modèles de machine learning. Ce langage possède un grand nombre de bibliothèques et de packages spécialisés pour l'analyse de données qui facilitent leur manipulation et leur traitement. Voici ci-dessous la liste des bibliothèques utilisées pour ce projet :

- **Numpy :** qui permet la manipulation de tableaux multidimensionnels et les calculs mathématiques.
- **Matplotlib :** qui permet la visualisation de données en 2D et 3D.
- **Seaborn :** qui permet la visualisation statistique de données.
- **Pandas :** qui permet la manipulation et l'analyse de données en tableaux.

2. Jeu de données

Le jeu de données à étudier est un fichier texte comportant 205 observations et 2 colonnes. Dans la première colonne nous avons les âges des canadiens observés et dans la deuxième colonne, nous avons le logarithme de leurs revenus. Ici, la variable explicative est l'âge et la variable cible est le logarithme du revenu.

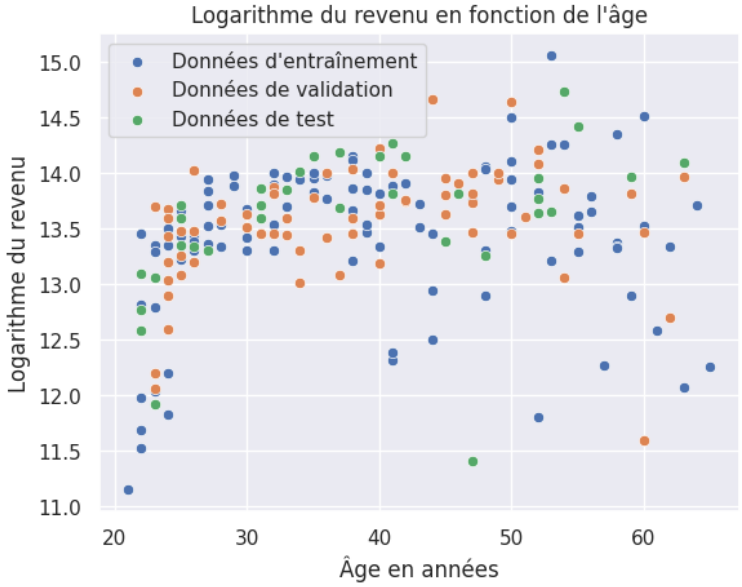
Nous devons estimer le modèle de ce jeu de données en appliquant une régression par processus gaussien. Cette régression permet de modéliser la distribution des valeurs de la variable de sortie (le logarithme du revenu d'un canadien) en fonction des valeurs de la variable explicative (l'âge). Cette distribution est décrite par des fonctions de covariance que nous présenterons plus tard dans ce rapport.

Pour pouvoir appliquer le processus gaussien, nous avons choisi de séparer aléatoirement le jeu de données en trois :

- Un jeu d'entraînement qui permet d'estimer les paramètres du processus gaussien, il représente 50% du jeu de données.
- Un jeu de validation qui permet d'optimiser les hyperparamètres θ et σ_{noise}, il représente 30% du jeu de données.
- Un jeu de test qui permet d'évaluer la performance du modèle, il représente 20% du jeu de données.

La répartition aléatoire a été fixé au début grâce à la fonction `fix_random_seed` avec le paramètre `seed` valant 42.

Voici ci-dessous le nuage de points représentant le jeu de données et sa répartition dans les différents jeux.



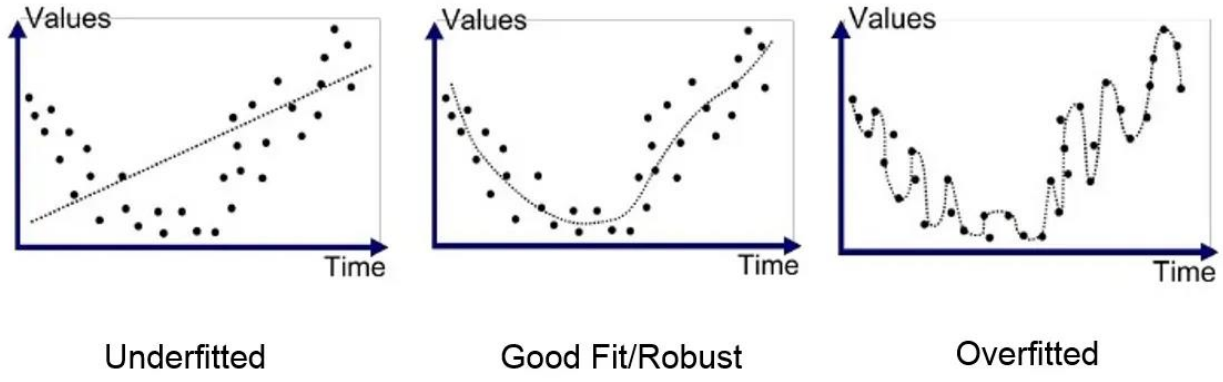
On observe que la répartition des données est uniforme.

3. Modélisation

Pour comparer l'efficacité des différents modèles, nous avons utilisé l'erreur quadratique moyenne :

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Aussi, nous avons fait le choix de fixer les hyperparamètres à 0,1 pour θ et 0,0001 pour σ_{noise}.



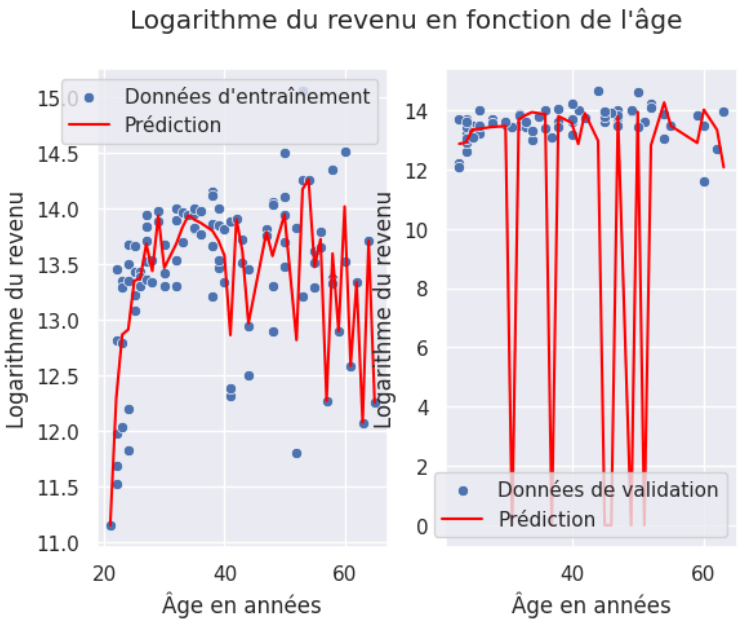
Pour chaque modèle, nous vérifierons si le modèle se trouve ou non dans un cas de sous ou sur apprentissage.

3.1. Analyse exploratoire

Voici ci-dessous la fonction que nous avons appliquée, la fonction covariance matern :

$$k(x,x') = \sigma^2\left(1+\frac{\sqrt{5}(x-x')}{\theta}+\frac{5(x-x')^2}{3\theta^2}\right)\exp\left(-\frac{\sqrt{5(x-x')}}{\theta}\right)$$

Après application des estimations des paramètres du processus gaussien et prédiction sur les jeux et d'entraînement et de validation, voici les résultats obtenus pour la fonction de covariance matern :



Nous pouvons observer que les prédictions sur les données d'entraînement comportent beaucoup de bruits. En effet, la courbe de prédiction fluctue énormément et « tente » de prédire beaucoup de données en collant les données d'entraînement. Cette remarque se confirme lorsqu'on observe la courbe de prédiction dans les données de validation. Le modèle comporte énormément de creux menant à aucune donnée réelle. À partir de ces observations, nous pouvons déduire que ce modèle de prédiction est en sur-apprentissage et qu'il n'est pas capable de se généraliser sur de nouvelles données.

Nous pouvons nous en assurer en comparant les erreurs quadratiques moyennes (EQM) du jeu d'entraînement et du jeu de validation. Nous obtenons les valeurs suivantes :

```
Erreur quadratique moyenne sur les données d'entraînement: 0.16581542337331287
Erreur quadratique moyenne sur les données de validation: 27.95813575770751
```

Nous obtenons pour le jeu de validation une EQM bien supérieure à celui du jeu d'entraînement. Cela signifie donc que nous sommes bien en présence d'un sur-apprentissage pour ce modèle de prédiction.

3.2. Tests des différentes fonctions de covariance

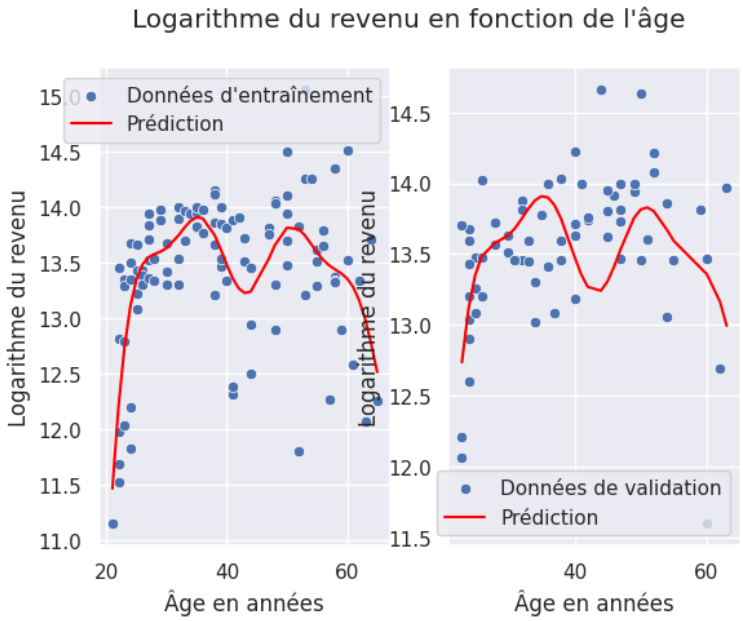
Nous allons appliquer d'autres fonctions pour comprendre si le problème de modélisation de la prédiction vient de la fonction choisie ou non.

3.2.1. Fonction gaussienne

Voici ci-dessous la fonction que nous avons appliquée, la fonction gaussienne :

$$k(x, x') = \exp(-\frac{(x - x')^2}{2\theta^2})$$

Après application du processus gaussien, voici les résultats obtenus :



Nous pouvons observer que la courbe de prédiction ne comporte très peu voire pas de bruit et qu'elle ne colle ni les données d'entraînement et de validation. La tendance principale de la courbe de prédiction est bien distinguable et capture une grande partie des données sans prendre en compte les données aberrantes. De cette observation, nous pouvons déduire que le modèle est fiable et capable de se généraliser sur de nouvelles données.

Nous observons que les EQM des données d'entraînement et de validation sont très proches. Nous pouvons ainsi conclure qu'il s'agit d'un bon modèle de prédiction.

```
Erreur quadratique moyenne sur les données d'entraînement: 0.270909732671763
Erreur quadratique moyenne sur les données de validation: 0.2480254449335573
```

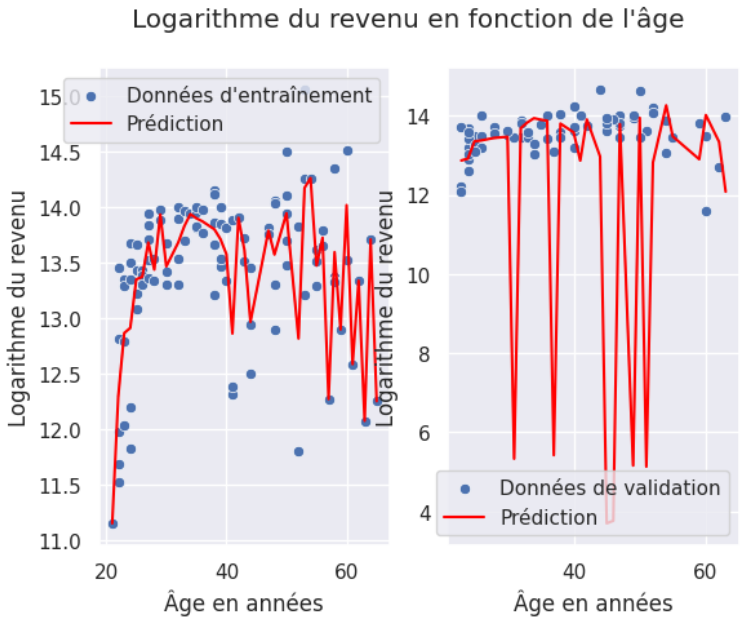
Nous obtenons pour les jeux d'entraînement et de validation, une EQM à peu près similaire. Cela signifie donc que nous avons là un modèle de prédiction fiable et capable de se généraliser sur de nouvelles données.

3.2.2. Fonction rationnelle quadratique

Voici-ci-dessous la fonction que nous avons appliquée, la fonction rationnelle quadratique (RQ) :

$$k(x, x') = 1 + \frac{(x - x')^2}{2\theta^2}$$

Après application du processus gaussien, voici les résultats :



Nous pouvons observer que, comme sur le tout premier modèle, les prédictions sur les données d'entraînement comportent beaucoup de bruits. En effet, la courbe de prédiction fluctue énormément et que la courbe de prédiction colle les données d'entraînement. Cette remarque se confirme lorsqu'on observe la courbe de prédiction dans les données de validation. Le modèle comporte énormément de creux menant à aucune donnée réelle. Nous pouvons, à partir de ces observations, déduire que ce modèle de prédiction est en sur-apprentissage et qu'il n'est pas capable de se généraliser sur de nouvelles données.

Nous pouvons vérifier cette déduction en vérifiant les EQM pour les données d'entraînement et de validation, dont voici les valeurs :

```
Erreur quadratique moyenne sur les données d'entraînement: 0.1658154214895222
Erreur quadratique moyenne sur les données de validation: 12.571020054567107
```

Nous obtenons pour le jeu de validation une EQM bien supérieure à celui du jeu d'entraînement. Cela signifie donc que nous sommes bien en présence d'un sur-apprentissage pour ce modèle de prédiction.

3.3. Optimisation des hyperparamètres

Pour améliorer les résultats, nous avons décidé d'utiliser la méthode de Maximum likelihood qui va nous permettre de procéder à l'optimisation des hyperparamètres θ et σ_{noise} qui maximisent la log-vraisemblance du processus gaussien définie par :

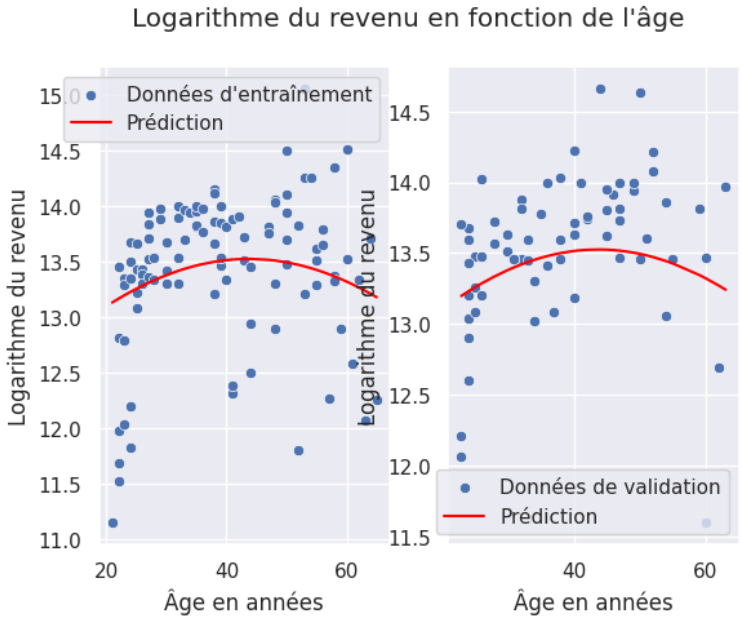
$$\log p(y|\theta, \sigma_{noise}) = \frac{1}{2} y^T K_y^{-1} y - \frac{1}{2} \log|K_y| - \frac{n}{2} \log(2\pi)$$

Où K_y est la matrice covariance bruité par σ_{noise} des données de validation (C. E. Rasmussen et C.K.I Williams, 2005).

Nous allons appliquer la log-vraisemblance aux trois fonctions vues précédemment.

3.3.1 Fonction matern :

Voici ce que nous observons en maximisant la log-vraisemblance :



Nous observons que la courbe de prédiction s'approche beaucoup plus des données pour le jeu de d'entraînement et de validation.

```
Erreur quadratique moyenne sur les données d'entraînement: 0.41000308661584495
Erreur quadratique moyenne sur les données de validation: 0.23559557841002932
```

Nous observons que les EQM sont très faibles et proches pour les deux jeux de données. Nous pouvons donc affirmer que ce modèle est fiable.

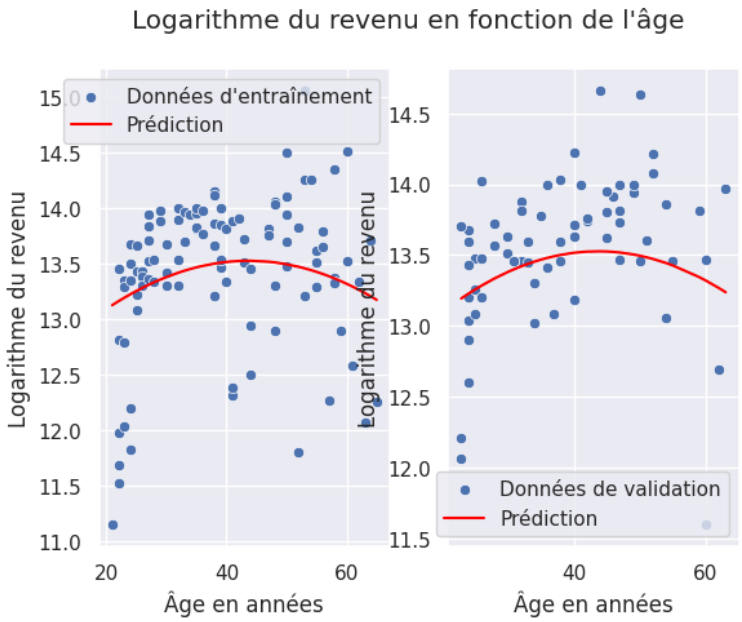
Voici ci-dessous les hyperparamètres avant et après optimisation :

```
Avant optimisation, les hyperparamètres sont:
theta = 0.1
sigma_noise = 1e-05
```

```
Après optimisation, les hyperparamètres sont:
theta: 119.76938056403648
sigma_noise: 0.6972697420103731
```

3.3.2 Fonction gaussienne :

Voici ce que nous observons en maximisant la log-vraisemblance :



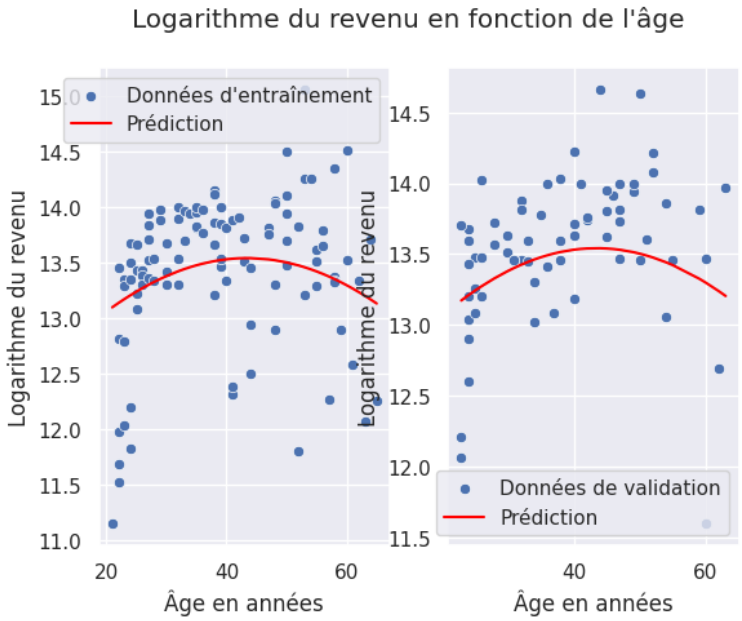
Ici, aussi nous observons que la courbe est proche des données. Le modèle sans log-vraisemblance était aussi fiable. La courbe de ce modèle est plus lisse.

```
Erreur quadratique moyenne sur les données d'entraînement: 0.40897734173651173
Erreur quadratique moyenne sur les données de validation: 0.2351065380607953
```

Nous observons que les EQM des deux jeux de données sont très faibles et proches. Nous pouvons affirmer que ce modèle est faible.

3.3.3 Fonction rationnelle quadratique :

Voici ce que nous observons en maximisant le maximum de la log-vraisemblance :



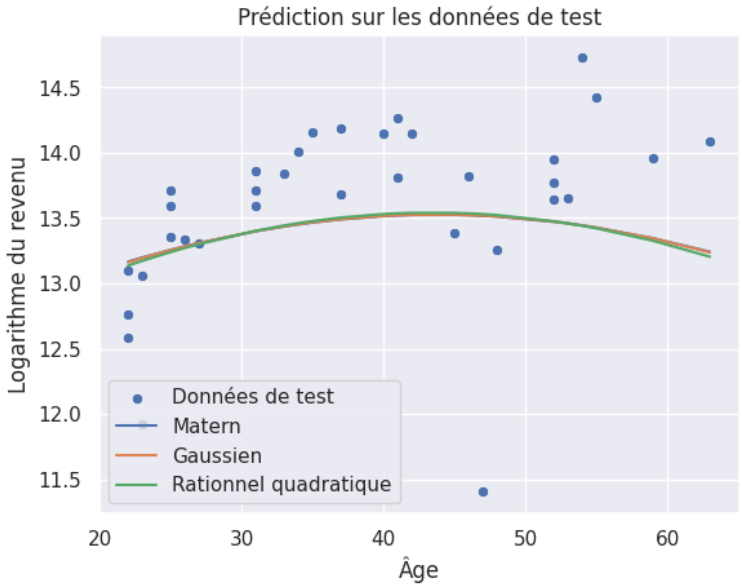
Ici également, nous observons que la courbe de prédiction s'approche beaucoup plus des données pour les jeux d'entraînement et de validation.

```
Erreur quadratique moyenne sur les données d'entraînement: 0.4037557801202349
Erreur quadratique moyenne sur les données de validation: 0.23101355882414254
```

Ici aussi, les EQM des jeux d'entraînement et de validation sont très faibles et proches, nous pouvons affirmer que ce modèle est fiable.

3.4. Comparaison des modèles sur le jeu de test

Ci-dessous, nous avons appliquer les trois modèles sur le jeu de test :



Nous observons que les trois modèles se superposent et qu'ils prédisent assez bien les données. Nous pouvons tout de même remarquer que la fonction rationnelle quadratique est la plus performante car son EQM est la plus petite, bien que la différence soit minime.

```
Erreur quadratique moyenne sur les données de test avec le noyau de Matern: 0.42675736250644736
Erreur quadratique moyenne sur les données de test avec le noyau gaussien: 0.42613618954456434
Erreur quadratique moyenne sur les données de test avec le noyau rationnel quadratique: 0.4244072444820659
```

4. Interprétation des résultats

L'interprétation de l'erreur quadratique moyenne peut s'effectuer à l'aide de sa racine carrée (RMSE) qui est de la même unité que la variable cible, ici le logarithme des revenues :

```
RMSE sur les données de test avec le noyau rationnel quadratique: 0.6236173068914604
```

La RMSE est d'environ 0,62 ce qui signifie que notre modèle prédit le logarithme du revenu à ± 0,62 près.

Voici ci-dessous l'intervalle de confiance à 95% de notre modèle :