Projet De Recherche
-
Volatilités Implicites et Pricing d'Options
-
Groupe 1

Acheroufkebir Yacine
Djigui Abiola Trésor
Ranger Alexandre

Juin 2020

Sommaire

1 Introduction

Les marchés financiers sont le lieu des échanges formels ou non de titres financiers, des matières premières entre différents individus qui représentent les acteurs du marché. L'évolution des marchés a conduit à l'apparition de nouveaux produits basés sur des actifs financiers sous-jacents: ce sont les produits dérivés. Pour mieux cerner ces marchés et opter pour une allocation optimale des ressources, toute une théorie visant à appliquer les mathématiques aux problèmes financiers s'est progressivement mise en place: les mathématiques financières. En mathématiques financières, il est très courant d'avoir affaire à des équations différentielles ou simples à résoudre pour déterminer des quantités nécessaires pour définir les caractéristiques du marché ou tout simplement pour en cerner la dynamique. L'un des problèmes de résolution couramment rencontrés est celui de la résolution d'équation différentielles partielles vérifiées par le prix d'un actif sur un marché.

Le prix d'un actif sur un marché financier suit un évolution aléatoire certes mais peut satisfaire une dynamique mathématique traduite au moyen d'une équation dite différentielle stochastique. Ainsi, le prix d'un actif sur le marché peut suivre une dynamique modélisable mathématiquement par des équations de Black-Scholes, de Heston, ou d'Orstein-Ulenbeck. Ces équations stochastiques peuvent au même titre que les équations différentielles ordinaires être résolues grâce à différentes méthodes numériques, par exemple: les méthodes de différences finies, les méthodes de Fourrier ou encore la méthode de Monte Carlo.

La connaissance de la dynamique de ces prix est utile pour déterminer à les prix d'options dérivant de ces actifs risqués: c'est le *pricing*. Les options dérivées les plus connues sont les Call et les Put. Ces instruments financiers qui donnent à leur détenteur un droit d'achat ou de vente doivent être "justement" pricés afin d'éviter des situations d'arbitrage sur le marché. Ainsi, pour un actif donné, et des caractéristiques de marché précises, il est possible de déterminer le prix d'un Call/Put via des méthodes numériques. Seulement, ces méthodes sont très coûteuses en temps. Avoir à les refaire à chaque fois qu'on veut pricer une option peut devenir très long. A l'heure où la rapidité des opérations est un facteur déterminant sur les marchés financier, il est primordial de trouver une méthode optimale pour le pricing de ces options. C'est pour cela que sont introduites les méthodes statistiques de l'IA.

Ces méthodes consistent à entrainer des modèles de pricing que l'on va calibrer sur des paramètres de marchés. Une fois ce modèle calibré, il pourra être utilisé off-line pour pricer de façon ultra-rapide des options connaissant les paramètres de marchés associés.

Le présent rapport décrit dans un premier temps, sans aller dans des détails profonds, la théorie autour des volatilités implicites, du pricing d'options et ses complexités temporelles d'un point de vue algorithmique. Il présente enfin l'apport

des méthodes statistiques d'abord au calcul des volatilités implicites et ensuite au pricing d'options.

2 Volatilités Implicites de Black-Scholes

2.1 Modèle de black-scholes

On considère un actif dont le prix S_t en fonction du temps satisfait l'équation:

$$dS_t = S_t(\mu dt + \sigma dB_t)$$

où μ, σ représentent respectivement le drift et la volatilité. W_t est un processus brownien. On dit que S_t suit un mouvement Brownien géométrique. Dans le cas où μ et σ sont des constantes, comme ce sera le cas dans nos travaux, on peut établir que:

$$S_t = S_0 exp((\mu - \frac{1}{2}\sigma^2)t + \sigma B_t)$$

.

Une option dérivée de type Call ou Put Européen a un payoff dépendant exclusivement de la valeur terminale du prix sous-jacent. Dans le cas d'un Call, la valeur du payoff est donnée par $p = (S_T - K)^+$. Le prix du Call associé est quant à lui donné à chaque instant $t \in [0, T]$ par $C_t = e^{-r(T-t)}\mathbf{E}[(S_T - K)^+ / F_t]$ où T représente la maturité, r le taux et K le strike. Distinguons deux facteurs : $(S_T - K)^+$ qui représente le payoff final et $e^{-r(T-t)}$ qui est le coefficient d'actualisation d'une valeur en T à la date t. Le prix du call se définit donc comme une actualisation à t de la valeur espérée du payoff final connaissant l'information à la date t. Remarquons que $\mathbf{E}$ est une espérance prise sous la probailité risque-neutre. On montre que le prix du call vérifie une équation partielle différentielle dîte de Black-Scholes:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0$$

où S représente le prix de l'actif sous-jacent à l'instant t. L'équation de Black-Scholes est souvent accompagné d'une condition terminale qui représente le payoff propre à l'option considéré. Dans notre cas, cette condition est:

$$C(t = T, S_T) = (S_T - K)^+$$

. On peut également montrer qu'une expression analytique de l'équation de Black-Scholes dans le cas d'une option Européenne est donnée par:

$$C(t, S) = SN(d_1) - Ke^{-r\tau}N(d_2)$$

où $\tau = T - t$, $d_1 = \frac{log(S/K)+(r-0.5\sigma^2)\tau}{\sigma\sqrt{\tau}}$ et $d_2 = d_1 - \sigma\sqrt{\tau}$.
L'idée de la preuve est décrite ci-dessous:

$$C(t, S_t) = F(t, S_t)$$

avec

$$F(t, x) = \mathbf{E}(e^{-r(T-t)}(x - K)^+)$$

Ici, S_t est un brownien géométrique, alors: $S_t = S_0 exp((\mu - \frac{1}{2}\sigma^2)t + \sigma B_t)$.

Sous la probabilité risque neutre considérée, on peut montrer qu'il existe un mouvement Brownien W_t tel que S_t a pour dynamique:

$$dS_t = S_t(rdt + \sigma dW_t)$$

Il s'en suit que:

$$S_t = S_0 exp((r - \frac{1}{2}\sigma^2)t + \sigma W_t)$$

.
Alors,

$$F(t, S_t) = \mathbf{E}(e^{-r(T-t)}(S_0 exp((r - \frac{1}{2}\sigma^2)t + \sigma W_t) - K)^+)$$

$$F(t, S_t) = \mathbf{E}[e^{-r(T-t)}(S_t exp((r - \frac{1}{2}\sigma^2)(T - t) + \sigma(W_T - W_t)) - K)^+]$$

$$F(t, S_t) = \mathbf{E}[(S_t e^{\sigma\sqrt{T-t}g - \frac{\sigma^2\theta}{2}} - Ke^{-r(T-t)})^+]$$

avec g une gaussienne centrée réduite.
On introduit les quantités $d_1 = \frac{log(S/K)+(r-0.5\sigma^2)\tau}{\sigma\sqrt{\tau}}$ et $d_2 = d_1 - \sigma\sqrt{\tau}$.
Avec ces notations, on peut écrire:

$$F(t, S_t) = \int_{-\infty}^{d_2} (S_t e^{\sigma\sqrt{T-t}g - \frac{\sigma^2(T-t)}{2}} - Ke^{-r(T-t)}) \frac{e^{-y^2/2}}{\sqrt{2\pi}} \, \mathrm{d}y$$

En écrivant cette intégrale comme la différence de deux intégrales et en faisant le changement de variable $z = y + \sigma\sqrt{T - t}$, on peut écrire:

$$C(t, S_t) = F(t, S_t) = S_t N(d_1) - Ke^{-r(T-t)} N(d_2)$$

2.2 Présentations et Calculs

Étant donné un prix d'option sur le marché V_m, et des paramètres S (prix du sous-jacent), K (Strike), τ (temps avant maturité), r (taux d'intérêt), on peut se demander la valeur σ^* de la volatilité à injecter dans la formule de pricing par Black-Scoles déterminée ci-dessus pour avoir $BS(\sigma^*, S, K, \tau, r) = V_m$. Cette quantité σ^* est appelée volaticité implicite.

Pour résoudre ce genre d'équation, on pourrait utiliser les méthodes de détermination des zéros de fonctions très connues en mathématiques: méthode de Brent, méthode de Newton ou encore la méthode de Bissection. Pour cela, il suffit de poser:

$$g(\sigma) = BS(\sigma, S, K, \tau, r) - V_m$$

où S, K, τ, r, V_m sont des constantes. Déterminer σ^* revient à chercher le(s) zéro(s) de la fonction g.

Nous pouvons, par exemple présenter la méthode de Newton-Raphson. Dans cette méthode, on cherche à construire une bonne approximation de la fonction g (supposée dérivable) en considérant son développement de Taylor au premier ordre. On obtient avec σ_0 un point de départ assez proche du zéro à trouver :

$$g(\sigma) = g(\sigma_0) + g'(\sigma_0)(\sigma - \sigma_0)$$

Pour trouver un zéro de la fonction g, on cherche à résoudre :

$$0 = g(\sigma_0) + g'(\sigma_0)(\sigma - \sigma_0)$$

On peut donc construire la suite récurrente :

$$\sigma_{k+1} = \sigma_k - \frac{g(\sigma_k)}{g'(\sigma_k)}$$

La méthode peut néanmoins échouer si à l'étape k σ_k n'appartient pas à l'ensemble de définition ou que la fonction n'est pas dérivable en σ_k. De plus, la valeur de départ ne doit pas être trop éloignée du zéro recherché sinon l'algorithme échoue. On peut montrer grâce à la formule de Taylor-Lagrange que :

$$|\sigma_0 - a| < K$$

avec $K = \frac{M_2}{2m_1}$ et $M_2 = \max(|g"(\sigma)|)$, $m_1 = \min(|g'(\sigma)|)$

Sous cette condition on obtient une convergence quadratique de la méthode. Cependant, on comprend que l'on doit contrôler le nombre d'itération par des conditions d'arrêt. Tout d'abord, on fixe un nombre maximal d'itération, nous choisissons $N = 10000$ et on arrête l'algorithme lorsque la différence entre deux termes de la suites des (σ_k) est inférieur à ϵ. Nous choisissons $\epsilon = 10^{-5}$.

On représente un exemple de la méthode de Newton-Raphson où l'on chercherait σ tel que $g(\sigma) = 0$

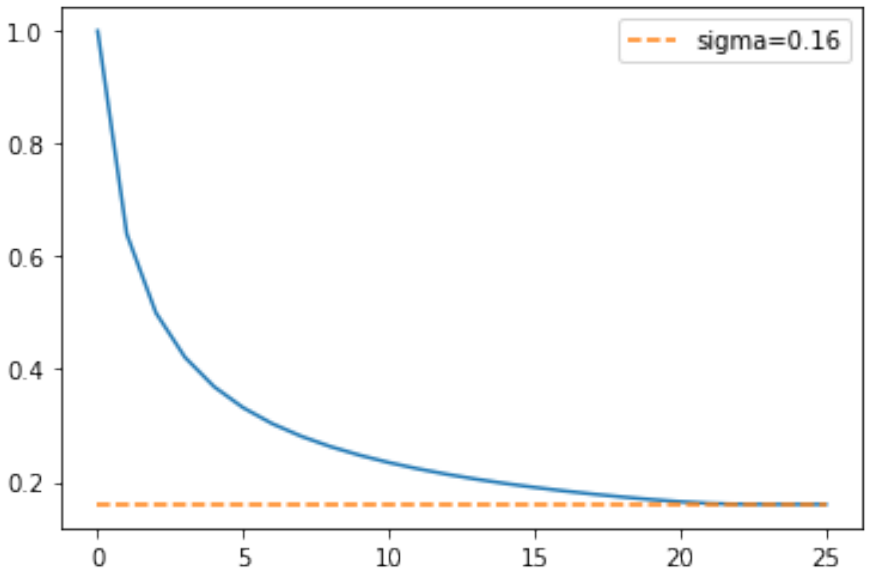

Figure 1: Newton-Raphson $\sigma_0 = 1$

On peut aisément remarquer la convergence quadratique vers $\sigma = 0.16$.

Cependant, ces méthodes toutes itératives peuvent poser des problèmes de stabilité ou de convergence voire même de complexité temporelle. Une solution serait donc de construire un modèle qui fait le lien entre le paramètre cible σ^* et les paramètres de marché. Une fois un tel modèle construit, cela permet des calculs très rapides, plus ou moins précis en fonction des pertinences du modèle.

Le paragraphe suivant présente un exemple de construction de modèle statistique appliqué au problème de détermination des volatilités implicites.

2.3 Exemple d'application des statistiques de l'IA au calcul des volatilités implicites

Il est possible d'utiliser des outils d'apprentissage automatique pour déterminer des volatilités implicites, l'avantage étant que ces outils sont facilement applicables et peuvent permettre de trouver des approximations plus simples à calculer.

Pour pouvoir utiliser ces outils, il faut travailler sur une base importante de données. Celle-ci sera générée en amont en prenant des valeurs aléatoires pour différents paramètres du marché. Or, connaissant la relation entre ces paramètres

et le prix de l'option du marché, on peut générer en utilisant la formule de Black-Scholes une colonne de prix. Voici la liste des paramètres et leurs bornes:

Paramètres	Bornes
K	[0.4 ;1.2]
S_0	[0.4;1.2]
x	[0.4 ;1.2]
σ	[0.01 ;1.0]
τ	[0.2 ;1.1]
r	[0.02 ;0.1]

En prenant donc les paramètres du marché en plus du prix de l'option sur le marché, on entraîne un modèle afin qu'il puisse donner la meilleure approximation possbile de la volatilité.

Soit un modèle $F(\theta)$ de paramètres $\theta \subset \Theta$ et $\psi \subset \Psi$ tel que Ψ est l'ensemble des données disponibles sur le marché. Tel que $F(\theta, \psi)$ donne une approximation de la volatilité dans le marché.

On note $S^{MKT}(\psi)$ la valeur réelle de la volatilité sur un marché défini par les paramètres ψ.

On pose également une fonction d'erreur notée ici

$$\delta : \Psi, F(\theta), S^{MKT} \longrightarrow \delta(\Psi, F(\theta), S^{MKT})$$

Où on a choisi de prendre le coefficient de détermination R^2.

On cherche donc pour le modèle choisi, le paramètre $\hat{theta}$ tel que :

$$\hat{\theta} = \underset{\theta \in \Theta}{argmin}\ \delta(\Psi, F(\theta), S^{MKT})$$

On considère alors que pour un modèle F, $F(\hat{\theta})$ est la meilleure approximation de S^{MKT}. Cependant il faut prendre en compte le fait qu'on utilise un nombre limité de données et que celles-ci ne sont potentiellement pas assez diverses pour traiter des cas extrêmes.

Cela peut entrainer du surapprentissage, ce qui réduit les capacités de prédiction pour les domaines en dehors des bornes de la base d'entrainement, c'est pour cela qu'il faut vérifier la validité du modèle sur une base test. C'est-à-dire qu'il faut comparer les scores des modèles sur des données qui n'ont pas été utilisées pour l'entrainement.

On étudie deux modèles prédictifs, l'arbre de régression et des réseaux de neuronnes. Ces modèles seront explicités dans la partie pricing. Ils ont été choisis car ils sont généralement plus flexibles et exigent peu d'hypothèses sur le modèle

réel. Pour vérifier que le modèle est juste on procède à une validation croisée en créant une base test sur laquelle on détermine le coefficient de determination. Un modèle est jugé satisfaisant si son coefficient de determination sur la base de test est proche de 1.

2.3.1 Arbre de regression

Une première régression est réalisée avec des arbres de régressions. Ces modèles sont simples et donnent souvent des résultats acceptables. Pour éviter le surapprentissage on limite le nombre de nœuds à 15 millions, valeur pour laquelle on observe les meilleurs résultats en terme de score sur la base test. Or ce score est relativement faible : le coefficient de détermination est de 0.788.

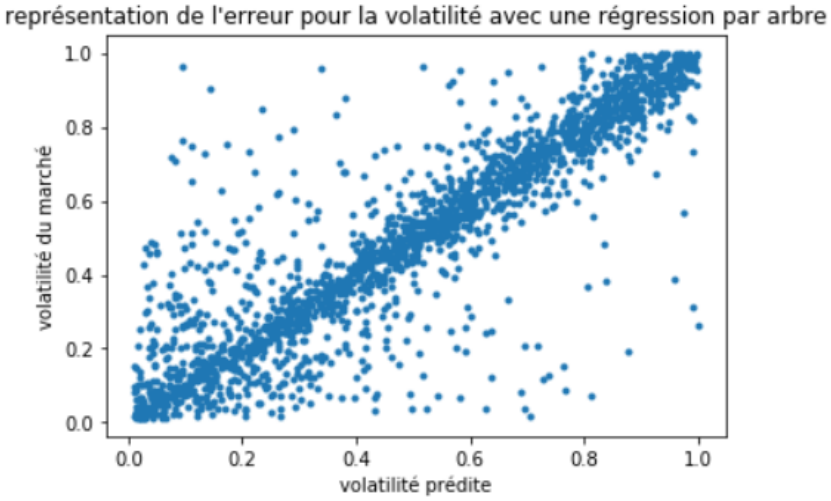

Figure 2: Arbre de décision

On observe une certaine correlation entre la volatilité réelle et celle prédite. Cependant, elle n'est pas suffisante pour justifier son utilité. Le but étant de pouvoir tracer une courbe de volatilité suivant les autres paramètres.

2.3.2 Réseaux de neurones

Le modèle des réseaux de neurones a donnée de meilleurs résultats que les arbres de régressions. Celui-ci a été paramétré avec 3 couches internes de 30 noeuds chacune. Le score est meilleur que le précédent, mais il est encore insatisfaisant. Le coefficient de détermination est de 0.9453379677849192.

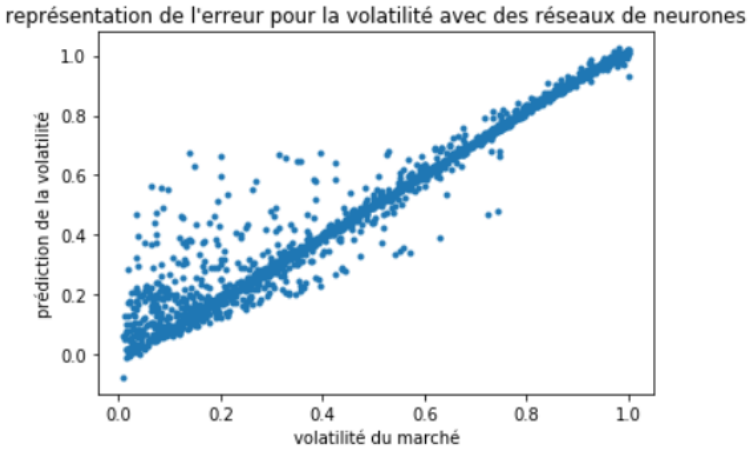

Figure 3: Réseaux de neurones

La courbe des volatilités prédites s'adapte mieux à celle des volatilités réelles que la méthode précédente. Cependant, l'erreur pour les petites valeurs de volatilité est encore trop importante. Ceci peut être expliqué par la taille relativement petite de la base par rapport au résultat souhaité.

Ici, on cherche un modèle qui est le plus proche possible de la réalité. Or nous n'avons pas les ressources nécessaires pour réaliser ce travail sur une telle base.

On obtient par ailleurs une nappe de volatilité correcte en fonction de la moneyness et de la maturité.

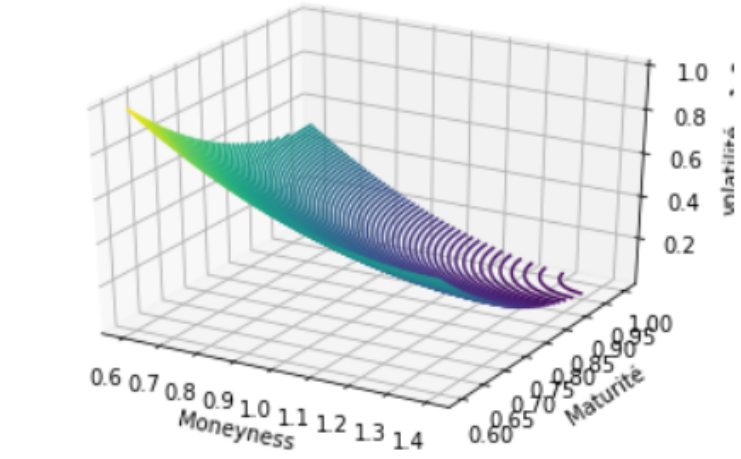

Figure 4: Nappe de volatilité pour K=0.65, x=0.5 et r=0.03

3 Pricing

Dans cette section, nous allons présenter le pricing d'un Call Européen via la méthode de Fourier et celle de Monte-Carlo. Nous allons ensuite présenter les complexités temporelles ainsi que les erreurs de précisions induites par ces méthodes. Enfin nous essaierons de calibrer des modèles statistiques qui permettent de pricer de façon beaucoup plus efficace.

3.1 Dynamiques des prix: Modèle de Heston

Nous supposerons cette fois que le prix de l'actif sous-jacent suit une dynamique de Heston, c'est-à-dire que le prix S_t vérifie:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{v_t} dW_t^s, S_{t0} = S_0 \\ dv_t &= \kappa(\bar{v} - v_t)dt + \sigma\sqrt{v_t} dW_t^v, v_{t0} = v_0 \\ dW_t^s dW_t^v &= \rho dt \end{aligned}$$

où v_t représente la volatilité instantanée, W_t^s et W_t^v deux processus browniens de coefficcient de corrélation ρ. L'équation traduisant la dynamique de v_t fait intervenir les paramètres suivants : r qui est le taux d'intérêt, $\bar{v}$ qui est la variance long-terme, κ qui est la vitesse d'inversion et σ qui représente la volatilité de la variance instantanée v_t. Enfin, V_0 désigne la valeur de v_t à l'instant t_0. Pour des raisons de praticité, on aura parfois à s'intéresser au log-prix du sous-jacent, défini par:

$$X_t = log(S_t)$$

On montre avec la formule d'Ito que la dynamique de Heston vis-à-vis du log-prix peut s'écrire:

$$\begin{aligned} dX_t &= (r - \frac{v_t}{2}) + \sqrt{v_t} dW_t^s, X_{t0} = log(S_0) \\ dv_t &= \kappa(\bar{v} - v_t)dt + \sigma\sqrt{v_t} dW_t^v, v_{t0} = v_0 \\ dW_t^s dW_t^v &= \rho dt \end{aligned}$$

3.2 Pricing d'options dérivées

Nous présenterons deux méthodes pour le prix à l'instant initial d'un call Européen: la méthode de Fourier et la méthode de Monte-Carlo.

3.2.1 Méthode de Fourier

Pour un Call Européen, le payoff final vaut:

$$p = f(X_T)$$

avec $f(x) = (x - K)^+$.
On définit le prix C_0 du Call à l'instant initial par:

$$C_0 = e^{-rT}\mathbf{E}[f(S_T)]$$

On définit les variables ci-dessous:

- p la loi de S_T
- $x = log(S_T)$
- ϕ_{X_t} est la fonction caractéristique de X_T sous la probabilité risque-neutre.
- $k = log(K)$
- $C(k)$ est le prix d'un call au temps t=0 pour une maturité T et un strike $K = e^k$

Posons $C_\alpha(k) = e^{\alpha k}C(k)$
On a que:

$$C(k) = e^{-rT}\mathbf{E}(S_T - k)^+$$

$$C(k) = e^{-rT}\int_k^{+\infty}(e^x - e^k)p(x)dx$$

On a:

$$C_\alpha(k) = e^{-rT}\int_k^{+\infty}(e^{x+\alpha k} - e^{(\alpha+1)k})p(x)dx$$

La transformation de Fourier de $C_\alpha(k)$ donne:

$$\tilde{C}_\alpha(\nu) = \int_{-\infty}^{+\infty} e^{i\nu k}C_\alpha(k)dk$$

$$= e^{-rT} \int_{-\infty}^{+\infty} \int_{k}^{+\infty} (e^{x+\alpha k} - e^{(\alpha+1)k}) p(x) dx dk$$

$$= e^{-rT} \int_{-\infty}^{+\infty} (\frac{e^{ix(\nu - i(\alpha+1))}}{\alpha + i\nu} - \frac{e^{ix(\nu - i(\alpha+1))}}{(\alpha + 1 + i\nu)}) p(x) dx$$

$$= e^{-rT} \phi(\nu - i(\alpha+1)) (\frac{1}{\alpha + i\nu} - \frac{1}{\alpha + 1 + i\nu})$$

$$= e^{-rT} \frac{\phi(\nu - i(\alpha+1))}{(\alpha + i\nu)(\alpha + 1 + i\nu)}$$

On en déduit donc l'esxpression suivante pour le prix du call:

$$C_\alpha(k) = \frac{e^{-rT-\alpha k}}{\pi} \int_0^\infty Re(\frac{\phi(\nu - i(\alpha+1))}{(\alpha + i\nu)(\alpha + 1 + i\nu)} e^{-i\nu k}) d\nu$$

En tronquant cette expression, on a:

$$C_\alpha(k) = \frac{e^{-rT-\alpha k}}{\pi} \int_0^L Re(\frac{\phi(\nu - i(\alpha+1))}{(\alpha + i\nu)(\alpha + 1 + i\nu)} e^{-i\nu k}) d\nu$$

Pour le modèle de Heston avec paramètres $\kappa, \bar{v}, \sigma, \rho$, on peut montrer que:

$$\phi(u) = \frac{exp(iulnS_0 + iurt + \frac{\kappa \bar{v} t(\kappa - i\rho\sigma u)}{\sigma^2})}{(cosh(\frac{\gamma t}{2}) + \frac{\kappa - i\rho\sigma u}{\gamma} sinh(\frac{\gamma t}{2}))^{\frac{2\kappa\bar{v}}{\sigma^2}}} exp(\frac{-(u^2 + iu)v_0}{\gamma coth(\frac{\gamma t}{2}) + \kappa - i\rho\sigma u})$$

Avec :

$$\gamma = \sqrt{\sigma^2(u^2 + iu) + (\kappa - i\rho\sigma u)^2}$$

Faisons la remarque qu'en fonction de α et L, le pricing par Fourier est plus ou moins précis. Ces précisions seront discutées dans les sections à suivre.

3.2.2 Méthode de Monte-Carlo

$$C_0 = e^{-rT} \mathbf{E}[f(S_T)]$$

Par Monte Carlo, on calcule $\mathbf{E}[f(S_T)$ en l'approchant par $\frac{\sum_{i=1}^N f(S_T^i)}{N}$ où les $(S_T^i)_{1 \leq i \leq N}$ sont des simulations indépendantes de S_T, prix terminal de l'actif sous-jacent.

Pour simuler S_T, on simule un log-prix $X_T = log(S_T)$ associé en utilisant la dynamique de ce dernier:

$$dX_t = (r - \frac{v_t}{2}) + \sqrt{v_t} dW_t^s, X_{t0} = log(S_0)$$

$$dv_t = \kappa(\bar{v} - v_t)dt + \sigma\sqrt{v_t} dW_t^v, v_{t0} = v_0$$

$$dW_t^s dW_t^v = \rho dt$$

On discrétise l'espace $[0, T]$ des temps de manière à avoir une subdivision $0 \leq t_0 < t_1 \cdots < t_{M-1} < t_M = T$ de pas constant Δt.
On utilise donc le schéma de simulation suivant:

$$X_0, v_0$$
$$\forall j \in [1; M]$$
$$\tilde{X}_{t_j} = \tilde{X}_{t_{j-1}} + (r - \frac{v_{t_{j-1}}}{2})\Delta t + \sqrt{\tilde{v}_{j-1}^+}\Delta W_j$$
$$\tilde{v}_{t_j} = \tilde{v}_{t_{j-1}} + \kappa(\bar{v} - v_{j-1})\Delta t + \sigma\sqrt{\tilde{v}_{j-1}^+}[\rho\Delta W_j + \sqrt{1-\rho^2}\Delta W_j^{'}]$$

où W et $W^{'}$ sont deux mouvements browniens indépendants. Rappelons que si (B_t) est un mouvement brownien standard comme c'est le cas pour W et $W^{'}$, ΔW est une gaussienne centrée de variance Δt.
Une fois que l'on simule assez de valeurs pour X_T, on peut en déduire les X_T correspondants puis estimer l'espérance dans l'expression de C_0. On obtient donc ainsi, après multiplication par le coefficient d'actualisation e^{-rT},le prix initial du Call.

3.3 Précisions et complexités de la méthode de Fourier

Les deux méthodes présentées plus haut sont en quelque sorte complémentaire. Les qualités de l'une complète les défauts de l'autre.
La méthode de Fourier a une précision principalement influencée par les paramètres α et L. Pour des paramètres de marchés fixés de manière à avoir un prix de call de valeur ~ 13.2023, nous faisons varier les paramètres α et L pour constater leur influence sur la précision de la méthode de Fourier. Les erreurs relatives pour quelques valeurs de α et L sont présentées dans le tableau ci-dessous:

α	**L=5**	**L=10**	**L=50**
0.1	0.0328	0.0282	$1.4322e^{-5}$
1	0.0341	0.0404	$2.1461e^{-5}$
20	1.4324	1.1903	0.0010

On constate que la méthode de Fourier est la plus précise pour les petits α et les grands L.
Par ailleurs, cette méthode a une complexité linéaire en $O(L)$, ce qui fait qu'elle est

très rapide lors d'un pricing.Par exemple, pour pricer 500000 Call via la méthode de Fourier, cela prend environ 17mins.
Malheureusement, cette méthode ne se comporte pas toujours bien dans certaines régions de valeurs des paramètres du marchés, ce qui fait de la méthode une méthode non universelle.

3.4 Précisions et complexités de la méthode de Monte-Carlo

La méthode de Monte-Carlo est très désavantageuse du point de vue de la complexité temporelle. En effet, elle a une complexité en temps de l'ordre de $M * N$ où M désigne le nombre de noeuds de l'espace des prix du sous-jacent après discrétisation et N le nombre de prix simulés pour approximer l'espérance. Mais pour avoir une très bonne précision il faut à la fois un pas de discrétisation minimal et aussi simuler un maximum de prix. Cela revient à un produit $M * N$ élevé et donc une complexité en temps non négligeable.
Il y a donc un arbitrage entre précision et complexité à prendre en compte.
Nous fixons N à 10 et pour différentes valeurs de M, nous nous intéressons au temps de simulation d'un prix via Monte-Carlo ainsi que l'erreur relative engendrée par la simulation. Les résultats sont réunis dans le tableau ci-dessous:

M	**temps d'exc**	**err. relative**
10	$3.1*10^{-3}$s	$\sim 23.27\%$
100	$3.1*10^{-2}$s	$\sim 3.1\%$
1000	0.25s	$\sim 0.51\%$
10000	2,4s	$\sim 0.19\%$
100000	24,4s	$\sim 0.12\%$

On remarque que la précision du résultat augmente considérablement avec M. Mais à l'opposé, le temps nécessaire est de plus en plus important lorsque M croit. L'avantage de cette méthode est qu'elle peut servir dans les régions de paramètres où la méthode de Fourier ne s'exécute pas convenablement.

3.5 Statistiques de l'IA et Pricing

Dans la partie 3.4, nous avons étudié deux méthodes de pricing des Call européens ainsi que la limite de ces méthodes d'un point de vue algorithmique. Dans la pratique, ce pricing se fait à très grande échelle ce qui pourrait faire exploser le temps de calcul des prix.

Pour contourner ce problème de lenteur, l'idée est d'entraîner un ou plusieurs modèles statistiques off-line puis de les utiliser pricer. A la fin, nous vérifierons trois indices principales:

- *rapidité* : les modèles entrainés off-line doivent nous permettre de pricer beaucoup plus vite que les méthodes de Fourrier et Monte-Carlo;
- *universalité* : contrairement à la méthode de Fourrier, les modèles statistiques entrainés doivent s'appliquer à toutes les régions des plages de paramètres retenues;
- *précision* : les modèles statistiques entrainés doivent avoir une précision avoisinant celle de la méthode de Fourrier voire mieux.

Si notre modèle final vérifie ces trois propriétés, nous pourrons alors le conseiller comme *pricer* en lieu et place des méthodes de Fourier et Monte-Carlo.

3.5.1 Base de données

La première étape correspondant à la mise en place des modèles statistiques est la génération de bases de données d'entrainement et de test.

Nous générons des paramètres de marchés (50000 de chaque) dans des plages prédéfinies précisées dans le tableau suivant:

Paramètres	**Plages**
Moneyness	[0.6;1.4]
Temps à maturité τ	[0.1;1.4](an)
Taux d'intérêt	[0.0;10](%)
Corrélation ρ	[-0.95;0.0]
Vitesse d'inversion κ	[0.0;2.0]
Variance long terme $\bar{v}$	[0.0;0.5]
Volatilité de la volatilité σ	[0.0;0.5]
Variance initiale v_0 σ	[0.05;0.5]

Ensuite, nous calculons les prix de Call associés aux paramètres de marchés définis par chacune des lignes des 50000 observations générées précedemment.

Pour calculer le prix associé à un n-uplet de paramètres, on utilise une fonction qui combine la méthode de Fourrier (à cause de sa rapidité) et la méthode de Monte-Carlo (car elle s'applique dans les régions où Fourrier ne s'applique pas). Le schéma de calcul de prix est décrit par la figure ci-dessous:

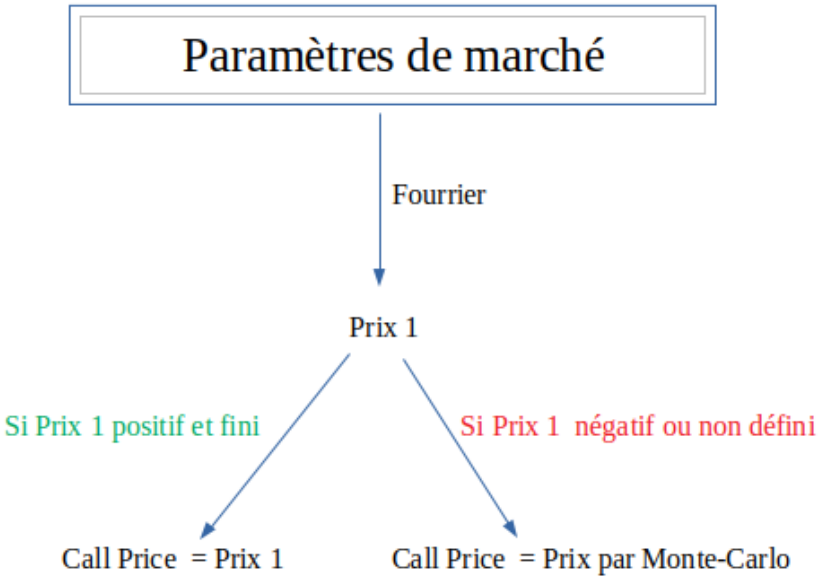

Figure 5: Schéma de Calcul des prix de Call lors de la création de la base de données.

Comme le montre le schéma ci-desssus, la méthode de Fourrier est prioritaire. Quand cette méthode renvoie un prix invraisemblable (négatif ou NaN), on fait intervenir la méthode de Monte-Carlo plus lente et moins précise mais qui a néanmoins le mérite de donner des prix plus vraisemblables.

L'avantage de combiner ces deux méthodes est d'obtenir une base de données sans déchets. Une base de données avec des prix négatifs nécessiterait une étape de nettoyage ce qui engendrerait la perte de certaines lignes d'observations et donc la réduction de la taille de notre base. Les paramètres α, L, N, et M sont respectivement fixés à 0.1, 50, 10 et 10000.

Enfin, une fois cette base de 50000 observations mise en place, elle a été divisée en deux: une base de données de 40000 observations qui serviront à l'entrainement des modèles et une base de 10000 observations qui serviront aux tests.

La figure suivante est un résumé de la base de donnée complète.

	kLog	S0	theta	V0	sigma	rho	r	kappa	tau	callPrice
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	5.000000e+04	50000.000000	50000.000000	50000.000000
mean	4.290903	99.595489	0.249731	0.299969	0.250239	-0.476105	4.957585e-02	1.000905	1.701439	44.729383
std	1.002782	57.618418	0.144779	0.144531	0.144574	0.274255	2.885907e-02	0.578151	0.404063	46.094227
min	-8.220824	0.031994	0.000004	0.050010	0.000003	-0.949961	8.154050e-07	0.000013	1.000016	0.000000
25%	3.903760	49.842611	0.122856	0.174712	0.124355	-0.713835	2.447263e-02	0.502060	1.350829	3.769585
50%	4.596093	99.027328	0.249438	0.299995	0.250712	-0.476516	4.937261e-02	1.000295	1.701202	30.709732
75%	5.003893	149.134135	0.375435	0.425157	0.375934	-0.238392	7.444095e-02	1.503788	2.052677	72.712269
max	5.298312	199.995235	0.499994	0.549986	0.499993	-0.000010	9.999936e-02	1.999962	2.399985	197.900608

Figure 6: Résumé de la base de donnée globale

3.5.2 Modèles linéaires

Une première étude consiste à supposer le modèle linéaire, et donc que la variable à prédire est par une approximation une combinaison linéaire des autres paramètres. La construction de ce modèle revient à chercher les coefficients de combinaison qui maximise la performannce. Pour y arriver, différentes méthodes s'offrent à nous.

Adoptons les notations suivantes:

$(X_i)_{1\leq 9}$ les vecteurs de variables explicatives, (Y) le vecteur de variable cible, et $(\beta_i)_{1\leq 9}$ les coefficients à déterminer. On veut établir:

$$Y \sim \sum_{i=1}^{9} \beta_i X_i$$

C'est à dire

$$\sim X\beta$$

où X est la matrice $(X_i)_{1\leq 9}$ de dimension 40000*9 et β le vecteur $(\beta_i)_{1\leq 9}$

Ordinary Least Square-OLS

Ici, on résout le problème suivant:

$$min_\beta \|Y - X\beta\|$$

On analyse la courbe des valeurs de prix prédits d'une base test par rapport à leur vraie valeur.

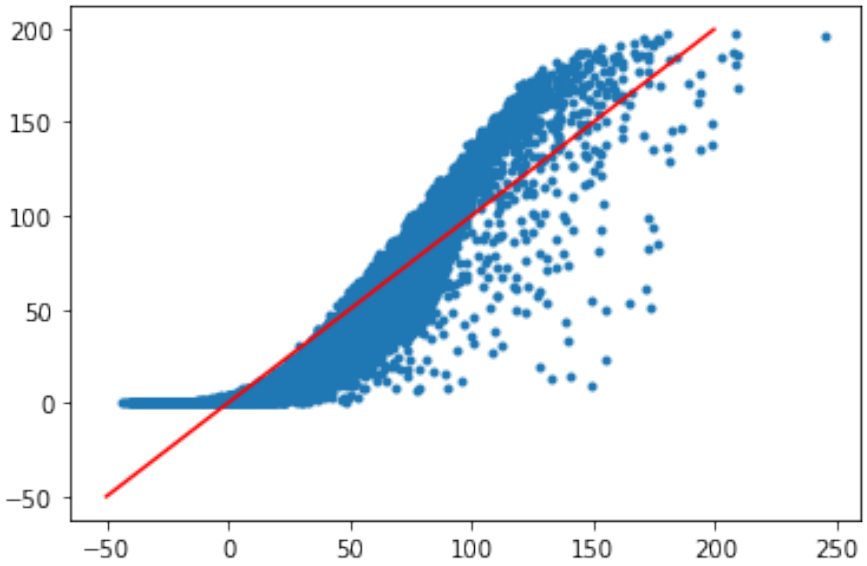

Figure 7: OLS : test: Observed VS Predicted

On remarque que la modélisation n'est pas du tout adapté au problème (score R-square ~ 0.85) et que des algorithmes plus développés vont être nécessaires.

Lasso

En raison du faible score à l'issu du modèle linéaire par OLS, nous allons explorer la méthode de Lasso qui comparativement à l'OLS fait intervenir un terme de pénalisation.

Ici, on cherche à résourdre le problème:

$$min_\beta \|Y - X\beta\| + \lambda\|\beta\|$$

avec $\lambda \in [0; 1]$

L'avantage de cette méthode par rapport à l'OLS classique est qu'elle peut faire office de sélection des variables en annulant certains coefficients β_i.

Nous mettons donc en place ce modèle en faisant varier le coefficient de pénalisation α et nous mesurons l'erreur quadratique moyen associée à chaque fois:

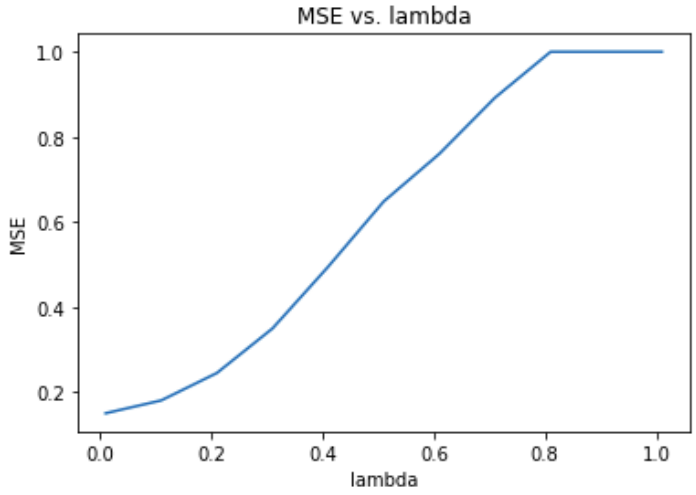

Figure 8: MSE fonction pour Lasso

On constate que la MSE croit en fonction de λ et varie entre 0.2 et 1 ce qui est une erreur moyenne énorme.

On constate que l'introduction d'un terme de pénalisation ne change rien à la donne: les indicateurs de performances restent faibles avec ou sans pénalisation. Nous allons donc étudier d'autres modèles.

3.5.3 k-NN : k-Nearest Neighbors

Une deuxième modélisation consiste en l'application de l'algorithme k-NN (k-Nearest Neighbors). Pour obtenir le nouveau prix en ayant les données initiales, on déduit la valeur comme la moyenne des valeurs des k plus proches voisins. L'importance du k est au centre de cette méthode.

Pour déduire le k, nous observons l'évolution du score, le score est la précision moyenne entre les résultats tests et les résultats retournés par l'algorithme. On décide de donner plus d'importance aux points plus proche du résultat à fournir. Dans notre cas, cela améliore la précision.

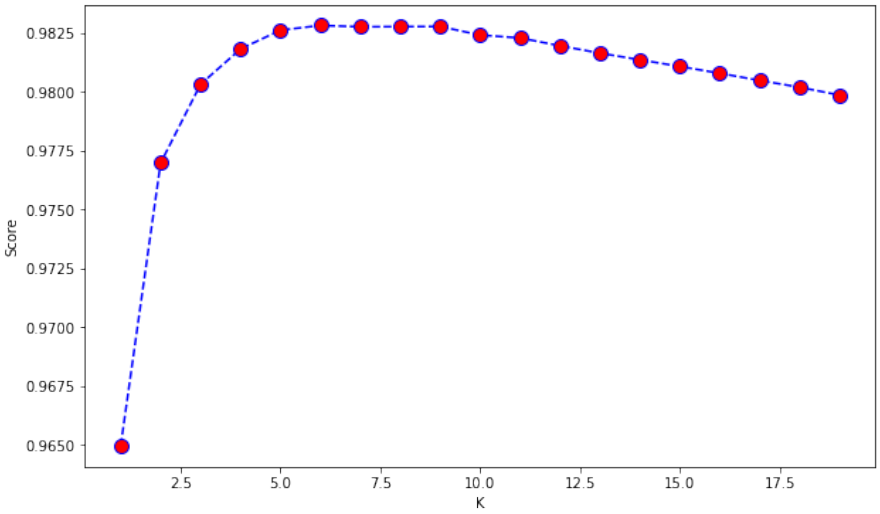

Figure 9: Évolution score vs k

D'après la figure précédente, on obtient le score maximal pour k=6, ça sera donc le nombre de voisins pris en compte pour en déduire le prix.

L'algorithme du k-NN nous donne, figure ci-dessous :

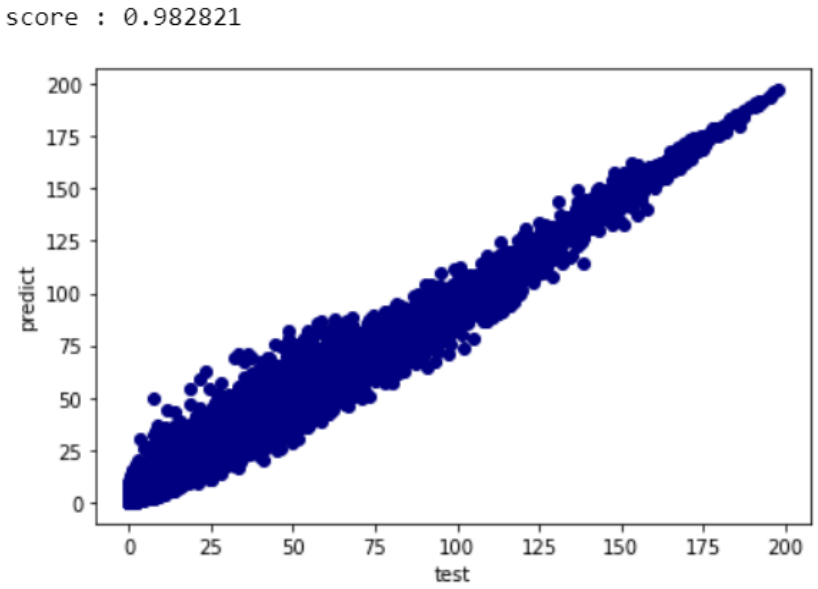

Figure 10: k-NN avec k=6

Nous obtenons un score d'environ 0.98 qui est assez correcte. Cependant, la dispersion est forte et la précision n'est pas bonne pour les petites valeurs. La

dispersion est assez forte, ce modèle n'est pas viable. On remarque que plus les prix augmentent plus la courbe s'affine et devient précise.

3.5.4 Arbres

Les arbres de décision sont des méthodes de régression flexibles, c'est à dire qu'ils ont tendance à adhérer le plus souvent aux observations quelle que soit la relation entre la cible et les valeurs en entrée. Ces modèles sont sensibles au surapprentissage, d'où la nécessité de travailler sur une large base d'entrainement.

Les algorithmes pour construire les arbres de décision sont construits en divisant l'arbre du sommet vers les feuilles en choisissant à chaque étape une variable d'entrée qui réalise le meilleur partage de l'ensemble d'objets. Pour choisir la variable de séparation sur un nœud, les algorithmes testent les différentes variables d'entrée possibles et sélectionnent celle qui maximisent la variance inter-classes.

Le modèle par arbre de régression a donné de meilleurs résultats que les précédents modèles. Avec un score de 0.9908765570215632 sur la base test.

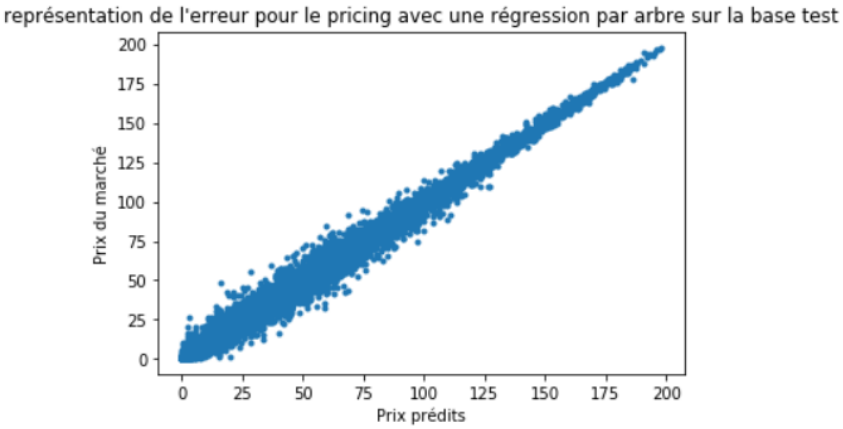

Figure 11: Arbre de décision

On observe encore que la dispersion est encore présente pour les petites valeurs, mais est moindre que pour le modèle du k-NN. Le modèle est cependant améliorable avec l'utilisation de modèles plus performants.

Nous avons par ailleurs également utilisé l'algorithme de boosting, qui s'apparente aux arbres dans le sens où les deux utilisent des classifieurs. Cette méthode est par contre difficile à paramétrer et nous n'avons pas réussi à obtenir un meilleur score que 0.9914957853712137. On a donc préféré tenter une dernière piste.

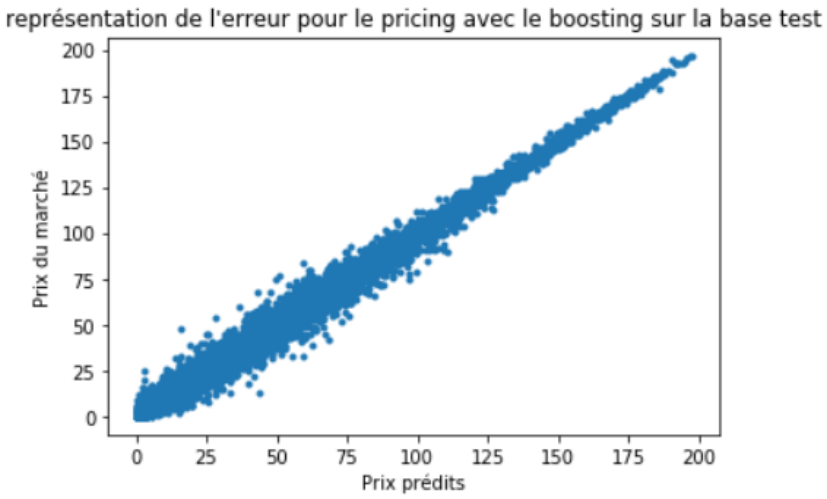

Figure 12: Boosting

3.5.5 Réseaux de neurones

Les réseaux de neurones artificiels sont des systèmes composés de neurones artificiels, qui reçoivent une entrée, ils combinent l'entrée avec leur état interne , un seuil défini à l'aide d'une fonction d'activation, et produisent une sortie. Les entrées initiales sont des données externes, tels que des images ou des documents. Les résultats ultimes accomplissent la tâche, comme la reconnaissance d'un objet dans une image

Le réseau de neurones utilisé a été organisé en plusieurs couches. Les neurones d'une couche se connectent uniquement aux neurones des couches précédentes et suivantes. Entre la couche d'entrée et celle de sortie peuvent se trouver plusieurs couches cachées.

Il existe une règle de base supplémentaire qui aide à résoudre les problèmes d'apprentissage supervisé. pour éviter un sur-apprentissage il faut garder le nombre de neurones en dessous de N_h:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o)}$$

$N_i = 9$ nombre de neurones en entrée
$N_o = 1$ nombre de neuronnes en sortie
$N_s = 40000$ la taille de la base d'entrainement
α un facteur arbitraire entre 2 et 10

On prends $\alpha = 10$ et on a donc pour le nombre de neurones $N_h = 400$ au plus répartis sur 4 couches.

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	(None, 9)	0
dense_17 (Dense)	(None, 100)	1000
dense_18 (Dense)	(None, 100)	10100
dense_19 (Dense)	(None, 100)	10100
dense_20 (Dense)	(None, 100)	10100
dense_21 (Dense)	(None, 1)	101

Total params: 31,401
Trainable params: 31,401
Non-trainable params: 0

Figure 13: paramètres des couches dan le réseau de neurones

Le résultat obtenu est bien meilleur que les précédents. On obtient un score de 0.9999425059354499. Ceci est observable sur la courbe suivante réalisée sur la base test.

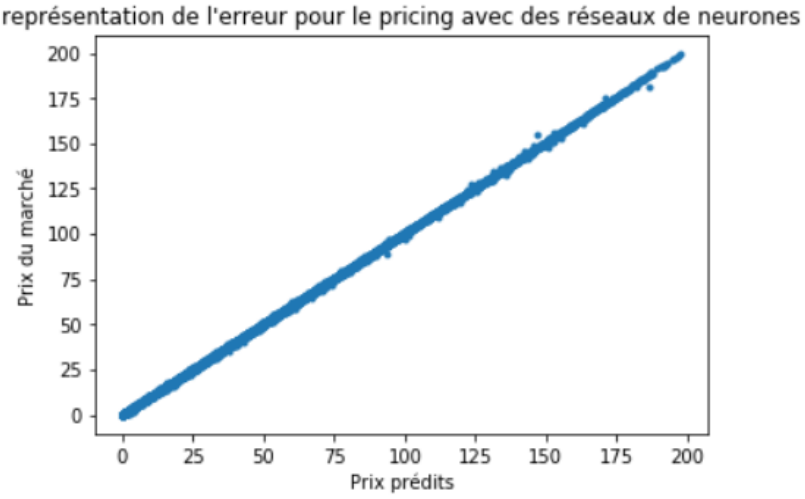

Figure 14: Réseau de neurones

La courbe des prix se confond parfaitement avec celle des prix prédits. Ce résultat est également meilleur pour les faibles valeurs de prix, pour lesquelles le coût des erreurs est plus important.

Ce modèle est satisfaisant et permet de tracer les courbes suivantes :

Les autres paramètres ont été fixés :

$$kLog = 3.5$$

$$S_0 = 30$$

$$\theta = 0.3$$

$$V_0 = 0.3$$

$$\sigma = 0.15$$

$$\rho = -0.15$$

$$r = 0.08$$

$$\kappa = 0.9$$

$$\tau = 1$$

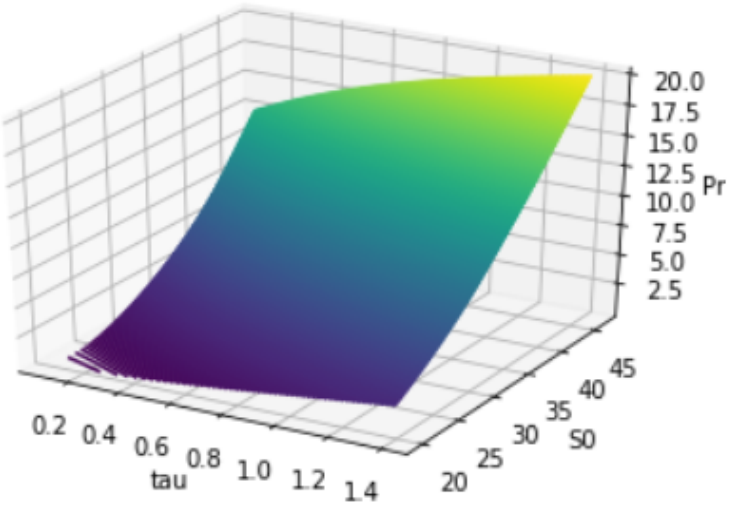

Figure 15: Prix du call en fonction de S_0 et de τ

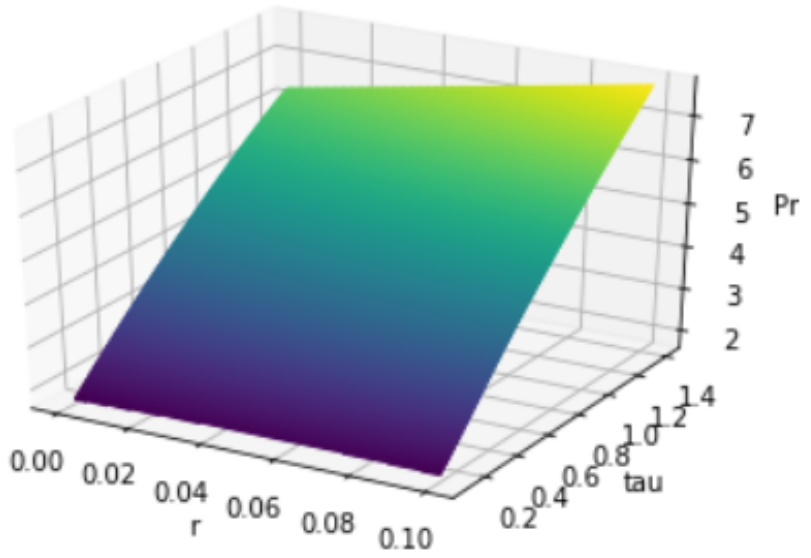

Figure 16: Prix du call en fonction de r et de τ

On observe que ces courbes correspondent bien aux observations sur le marché. Comme présenté ici, le Prix est croissant avec S0 et décroit par rapport à la maturité. Il faut rappeler qu'ici $\tau = T - t$.

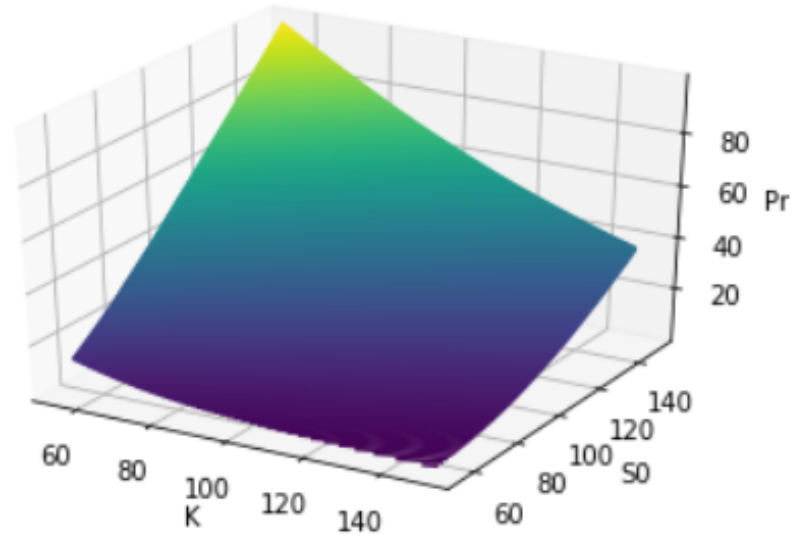

Figure 17: Prix du call en fonction de K et de S_0 pour $\tau = 1$

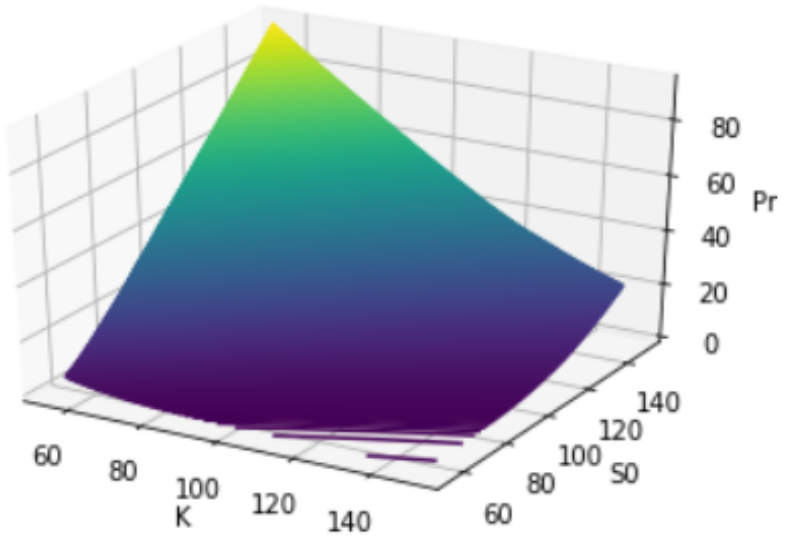

Figure 18: Prix du call en fonction de K et de S_0 pour $\tau = 0.1$

De même on observe sur ces deux dernières courbes que le prix est décroissant par rapport au strike, mais aussi qu'il y a un écrasement de la nappe de prix, plus on s'approche de la maturité.

4 Conclusions

L'utilisation des outils statistiques et d'IA s'est avéré être une bonne alternative dans certains cas. L'ordre d'acceptabilité des modèles étudiés ci-dessus est: Modèle linéraire (0.84 de score), Modèle par Knn (0.98 de score), Modèle par arbre (0.9914 de score) et les réseaux de neurones (0.9999 de score).

L'avantage de ces modèles est qu'ils recourent à des calculs plus simples et plus rapides que si on utilisait les méthodes de Fourier ou de Monte Carlo pour calculer le prix à chaque fois. En plus, la précision d'approximation offerte par certaines d'entre elles (score ≥ 0.98) est largement acceptable.

Ainsi, une fois que l'on a passé une phase d'apprentissage très coûteuse en temps, ces modèles statistiques permettent de pricer de manière ultra-rapide quand on a affaire à du pricing à grande échelle. Si le calcul de 10000 prix prend environ 17mins pour s'effectuer via la méthode de Fourrier, il ne prend que quelques secondes avec les réseaux de neurones. En plus, les réseaux de neurones calculent correctement les prix dans les régions où la méthode de Fourrier a dû mal.

Cependant, la création de ces modèles dépend d'hypothèses et du choix des bon paramètres sur la base d'entrainement. La pertinence de ces modèles n'est donc pas assurée pour des paramètres de marché qui ne sont pas dans les domaines de paramètres considérés pour l'entraînement. On a pu le remarquer pour la volatilité implicite.