ECE PARIS · LYON
ÉCOLE D'INGÉNIEURS

Rapport Final

Projet de Fin d'Étude PFE
Dynamic Pricing Strategy

Optimus Price

Arnaud BAZIN de BEZONS
Sylvain BENISTAND
Jonathan BOUTAKHOT
Loïc CHEONG
Thomas HU
Cécile WANG

Mentor : Jae Yun JUN KIM

ECE PARIS
2020-2021

ECE PARIS · LYON
ÉCOLE D'INGÉNIEURS

SOMMAIRE

I. Introduction & Contexte

Tout au long de ce rapport nous allons vous présenter notre projet de fin d'études, réalisé lors de notre dernière année à l'ECE Paris. Nous avons pris connaissance de ce projet suite à la lecture du sujet proposé par notre professeur et mentor Monsieur Jae Yun JUN KIM. Et c'est avec une certaine curiosité et une envie d'apprendre dans cette partie très importante du commerce qu'est la fixation des prix que nous l'avons choisi.

Le projet que nous avons réalisé, Optimus Price, est une réalisation technique qui tente de modéliser l'interaction entre un vendeur intelligent et un groupe de clients dans l'objectif de maximiser les profits du vendeur. Pour ce faire, l'objectif final était de mettre en place un dynamic pricing. Cela consiste à adapter le prix des produits en fonction de plusieurs facteurs qui sont internes ou externes à l'entreprise.

Le dynamic pricing est appliqué et applicable dans de nombreux domaines et pour de multiples produits; nous le trouvons principalement dans la fixation des prix de billets d'avion ou encore la gestion de revenus dans l'hôtellerie.

Nous avons fait le choix d'utiliser les données d'Airbnb car elles sont facilement trouvables, claires, et beaucoup d'exemples d'utilisation de celles-ci sont disponibles en ligne. Il y a également la notion de limite dans le temps, car il y a une période limite de réservation et de "stock" pour chaque mois, cette notion de temps est un facteur dans l'adaptation du prix. Cela permet d'observer la perspicacité et la précision de notre modèle sur une application de la vie courante.

L'exemple d'Airbnb peut être remplacé par n'importe quel type de produit ayant une maturité de temps et une limite de stock. Les seules adaptations à faire se feront au niveau des données à transmettre pour la fixation du prix du côté du vendeur et certains concepts à adapter dans la modélisation des clients. Nous verrons par la suite que nous avions comme objectif de modéliser deux types de clients que nous décrirons précisément.

Les autres demandes venant du mentor reposent principalement sur la modularité du code ainsi qu'une interaction claire entre le vendeur intelligent et les clients.

La réalisation de ce projet s'est faite grâce à l'étroite collaboration de 2 parties prenantes:

Tout d'abord notre mentor, Monsieur Jae Yun JUN KIM qui, en plus d'avoir proposé ce projet, nous a accompagné lors du démarrage de celui-ci pour la recherche d'articles scientifiques, la compréhension du Q-learning ainsi que l'apprentissage par renforcement. Puis par la suite, tout au long des réunions, il nous a guidé dans la compréhension et la vision pour aborder les différentes problématiques auxquelles nous faisions face durant la réalisation du projet.

Et ensuite, nous, les étudiants qui nous sommes totalement impliqués dans la réalisation d'Optimus Price. Nous sommes une équipe de 6 étudiants dont 4 en majeure Finance et 2 en majeure Système d'Information. Cette mixité permet d'avoir de solides connaissances en informatique appliquées à de multiples facettes de cette discipline allant de la gestion et du traitement de données jusqu'au Machine Learning.

II. Problématique

Il existe de plus en plus de e-commerce dans le monde, la compétition est intense, de plus avec la situation actuelle, les clients achètent plus régulièrement en ligne. Il est donc important pour les vendeurs d'offrir leurs produits au meilleur prix. Cette recherche du meilleur prix est bénéfique autant pour le vendeur [1]que pour le client, le but étant de gagner un maximum de parts de marché et d'écouler leur stock.

La détermination du prix dépend de plusieurs facteurs notamment de la relation entre l'offre et la demande, la maturité du produit, la quantité disponible, la présence de concurrents, l'état du marché ainsi que le comportement des clients qu'ils soient naïfs ou stratégiques.

La problématique qui se pose alors est la suivante :

Quelle stratégie de dynamic pricing à adopter dans un marché en présence de clients stratégiques avec un produit ayant une maturité fixée (finie ou infinie) ?

Dans la suite de ce rapport nous tâcherons de décrire les étapes et les résultats qui nous ont permis de répondre à cette problématique.

III. Etat de l'art & Concurrence

A. L'existant

Le dynamic pricing est souvent utilisé par certaines entreprises pour optimiser le prix d'un produit tout en restant compétitif sur le marché. Les exemples les plus connus sont les compagnies aériennes et les services hôteliers.

Pour entamer notre projet, nous avons effectué des recherches d'articles scientifiques pour analyser ce qui avait déjà été fait et ce qu'il reste à faire. Dans notre cas, nous nous sommes concentrés sur l'utilisation de l'apprentissage par renforcement et plus précisément par le Deep-Q-Network (DQN). En effet, cela était une demande explicite de notre mentor.

En plus de l'utilisation du DQN, nous avons également fait des recherches sur le comportement des clients vis-à-vis du prix des produits afin de cerner au mieux les comportements de ceux-ci. Enfin, nous nous sommes attardés sur l'influence de la maturité des produits car c'est également un facteur à prendre en compte dans notre projet.

L'article *Event Ticket Price Prediction with Deep Neural Network on Spatial-Temporal Sparse Data* met en avant l'utilisation du Dynamic Pricing pour la vente de tickets pour des événements. Cet article insiste sur la différence entre la détermination du prix d'un événement et celui d'un billet d'avion. En effet, concernant les billets d'avion les facteurs à considérer sont le nombre limité de places ainsi que l'échéance du vol, tandis que pour un ticket d'événement il faut prendre en compte le choix des places mais également le caractère unique de l'événement qui influe le comportement des clients. La présence de différents facteurs favorise l'utilisation d'un réseau de neurones d'après l'article.

Reinforcement learning for fair dynamic pricing insiste sur l'avantage d'utiliser de l'apprentissage par renforcement pour maximiser le prix tout en conservant un équilibre entre le profit et un prix considéré juste au yeux des clients. Pour comprendre le comportement des clients, il faut distinguer les clients naïfs des clients stratégiques. Un client naïf est caractérisé par un prix unique, si le produit est en dessous de ce prix le produit est acheté sinon le client quitte le marché. En revanche, un client stratégique est caractérisé par le prix car il veut minimiser ses dépenses. Ce type de client peut avoir d'autres critères de choix comme la qualité par exemple.

Selon l'article *Stochastic Dynamic Pricing with Strategic Customers and Reference Price Effects* le client stratégique se caractérise par son *Willingness to pay* c'est-à-dire le prix qu'il est prêt à dépenser pour le produit et le temps qu'il est prêt à attendre pour que le prix du produit baisse. Le client analyse alors le marché, afin de se positionner au mieux dans le marché.

Nous avons donc constaté qu'avec le choix de notre produit (un appartement mis en location sur Airbnb), il est indispensable d'utiliser l'apprentissage par renforcement à travers un réseau de neurones pour une meilleure gestion des données ainsi qu'une meilleure précision de données de sortie.

B. Concurrence

Sur le marché actuel il existe plusieurs logiciels de veille pour E-commerce plus sophistiqués les uns que les autres. Dans un premier temps il y a *Omnia Retail* qui permet de faire une étude du marché , une étude marketing et de la gestion de stock pour la fixation du prix. *Price2Spy* est également sur le marché, il offre en plus d'un prix dynamique un rapport de synthèse du marché. Enfin, on peut compter la société *LiquidPrice* qui propose les mêmes services que ses concurrents pour optimiser les prix tout en optimisant le stock.

Les logiciels existants sont coûteux et appropriés aussi bien pour les grandes entreprises que pour les petites/moyennes et ont toujours l'objectif annexe d'optimiser les stocks.

Nous proposons une solution utilisable pour tous et dont le but est de maximiser les bénéfices générés pour un produit avec un horizon de temps fini en prenant en compte le comportement des clients.

Logiciel	Dynamic Price	Tous types d'entreprise	Analyse interne/externe	Clients stratégiques
Net Rivals	✔	✔	✔	✗
Omina Retails	✔	✗	✔	✗
Price2Spy	✔	✗	✔	✗
Optimus Price	✔	✔	✔	✔

IV. Description de l'approche méthodologique

Après 5 années à l'ECE nous le savons, la bonne réussite d'un projet réside avant tout sur l'organisation. En début de projet nous avons donc créé le diagramme de Gantt suivant afin d'estimer au mieux le chemin critique auquel nous devions prêter toute notre attention :

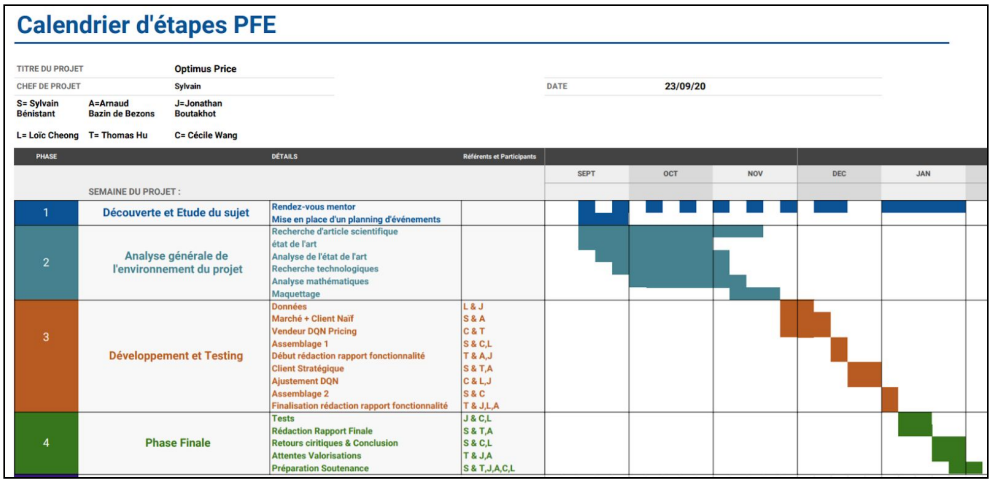


Diagramme de Gantt

Conscients de cela une fois notre rapport de recherche réalisé et les possibles difficultés identifiées, nous nous sommes mis en recherche de données. Nous en reparlerons dans la section difficultés(partie VII) mais nous avions initialement choisi de traiter des prix de billets de concerts ou d'évènements sportifs. Après 2 mois de recherche sans résultat nous avons préféré bifurquer vers le pricing d'un autre produit : les appartements en location sur AirBNB.

A. Les outils

Pour réaliser ce projet, nous avons exclusivement développé nos programmes en langage Python. C'est le langage de programmation privilégié et le plus utilisé dans le monde dans le domaine du machine learning ce qui est un point très important car le nombre de ressources disponibles sur Internet est très conséquent.

Nous avons utilisé Spyder (IDE de Python) et parfois Google Colab pour développer nos programmes en Python. C'est un langage relativement simple à apprendre et avec une forte lisibilité de compréhension. Dans le domaine des sciences des données, l'analyse s'effectue beaucoup plus rapidement avec une manipulation aisée et efficace des données avec la librairie Pandas. On peut lire et écrire les données structurées en gérant de gros volumes de données. Dans notre projet, cela était de l'ordre de 3 millions de lignes avec 18 colonnes. Ceci est beaucoup plus pratique que de les manipuler avec une table Numpy, où il aurait fallu utiliser une boucle *for* pour parcourir toutes les données. La complexité algorithmique serait alors linéaire *O(n)*. Ce qui signifie que le temps de calcul sera long, ce qui n'est pas approprié sur une grande quantité de données. On peut également traiter et accéder plus rapidement les données en utilisant les index en tant que chaînes de caractères, effectuer des tris, redimensionner et fusionner des tables.

Nous avons codé un Deep Q-Networks (DQN) grâce à la bibliothèque PyTorch pour *pricer* les logements Airbnb afin de maximiser le profit de leurs propriétaires. Le DQN est un algorithme de Reinforcement Learning (RL) combinant le Q-learning et des réseaux de neurones profonds (Deep Neural Networks) permettant au RL de fonctionner dans des environnements complexes et de grande dimension. Cela est moins gourmand en temps de calcul et d'exécution. Nous détaillerons le fonctionnement du DQN dans la suite de ce document.

Durant notre phase de recherche, la majorité des codes sur le DQN utilisaient PyToch. Nous avons alors privilégié cette même librairie plutôt que TensorFlow pour mieux appréhender le fonctionnement du DQN.

B. Les données

Concernant les données d'Airbnb qui sont disponibles et libres d'accès, nous avons dans un premier temps sélectionné les données qui, pour nous, sont considérées comme utiles et utilisables pour le modèle.

Nous avons donc sélectionné les données de New-York (E-U) car c'est le set où les données sont les plus anciennes et où elles sont surtout les plus nombreuses. En effet, après avoir filtré sur le type de biens mis en location, nous avons décidé de ne conserver que les données des appartements entiers ce qui représente un total d'1.5 millions d'appartements listés au moins une fois depuis 2015.

Afin de conserver un cohérence pour le modèle, nous avons ensuite filtré les appartements en fonction de leurs prix et de ceux qui avaient été sur le site de 2015. Nous obtenons alors 504 appartements avec un prix compris entre 100 et 200$ avec les informations mensuelles depuis 2015.

Comme on peut le voir dans les graphiques ci-dessous, les prix moyens sont assez stables et le niveau de disponibilité moyen est cyclique avec une petite augmentation lors du premier semestre 2020 dûe à la crise.

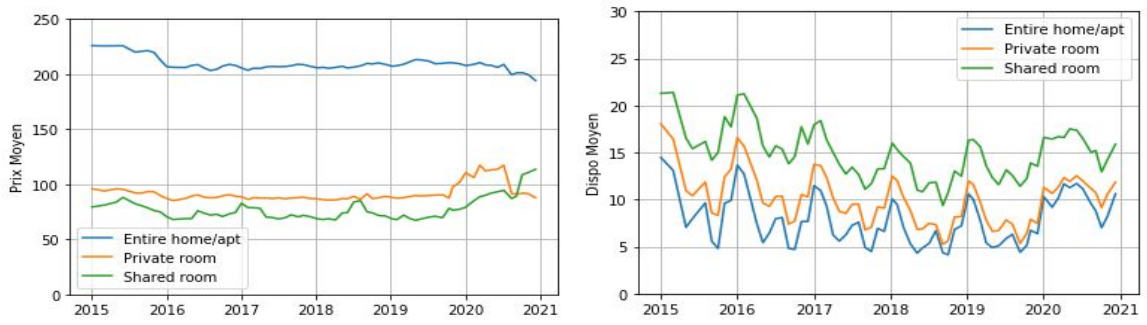


Schéma des Prix moyens et disponibilités des AirBNB de New-York

Ces données sont ensuite utilisées comme base à la modélisation du comportement des clients détaillée dans les prochaines parties.

V. Solution

A. Côté Vendeurs

1. Explication Q-learning

Le Q learning est une technique d'apprentissage par renforcement. C'est un domaine du machine learning consistant à laisser un agent interagir dans un environnement au cours du temps, où son but est d'apprendre les meilleures actions à prendre afin d'optimiser une récompense à long terme. Pour cela, l'agent se base sur l'état de l'environnement et sur ses expériences passées. Pour chaque action prise à un état donné de l'environnement, celui-ci donne une récompense à l'agent, positive ou négative. Ainsi, l'agent cherche un comportement optimal, appelé stratégie ou politique, en maximisant la somme des récompenses au cours du temps.

L'objectif du Q-learning est de trouver la politique optimale de sorte que la récompense totale espérée sur toutes les étapes successives soit un maximum atteignable.

La politique est la probabilité que l' agent choisisse une certaine action pour un état donné. Etant donné le peu d'informations lors des premières étapes pour choisir une action, il est possible de mettre en place une « stratégie cupide d'epsilon » (epsilon greedy strategy). Cette stratégie guide l'agent à choisir des actions selon deux axes : l'exploration et l'exploitation.

- Epsilon = 1, exploration des données, de l'environnement

- Epsilon = 0, exploitation des données, des expériences passées

Il est en général conseillé de fixer ce taux epsilon à 1 au début de sorte que l'agent explore tous les champs des possibles en vue de les réutiliser pour optimiser ses prochaines actions.

La fonction de valeur représente la valeur d'une certaine action ou certain état pour un agent. La fonction de valeur d'action (action-value function) est appelée Q-function, le résultat qu'il renvoie est la Q-value. Ces valeurs peuvent être enregistrées dans un tableau, communément appelé la Q-table, pour toutes les combinaisons possibles d'état et d'action prise à cet état. La Q-function deviendra optimale par itération grâce à l'équation de Bellman. Il s'agit d'une équation nécessaire pour atteindre l'optimalité. Pour chaque itération, une nouvelle Q-value est calculée, supposée converger vers la Q-value optimale. Elle est stockée dans la Q-table par-dessus l'ancienne valeur et la notion de taux d'apprentissage entre en jeu (learning rate alpha). Ce facteur joue sur l'importance entre les valeurs passées et les valeurs présentes, en d'autres termes il correspond à la pondération de la Q-value entre chaque itération.

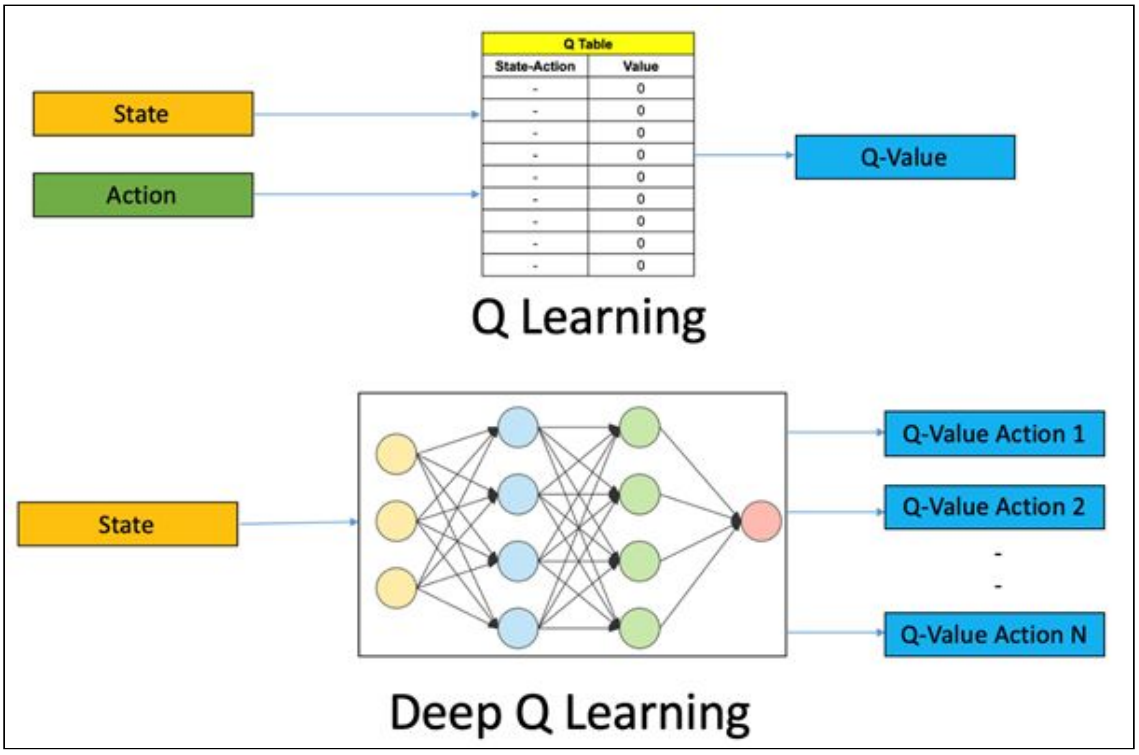
- Alpha = 1, signifie la prise en compte seulement de la nouvelle valeur calculée et par conséquent l'ignorance totale des anciennes valeurs.

- Alpha = 0, signifie donc la prise en compte des anciennes valeurs seules et non de la nouvelle valeur calculée.

Pour récapituler, en Q-learning nous utilisons l'équation de Bellman pour faire converger la Q-function vers la Q-function optimale. Avec cette Q-function optimale, nous déterminons la politique optimale qui décide de la séquence d'actions optimale à prendre pour maximiser la récompense cumulée au cours des étapes.

2. Explication Deep Q-learning

Lors d'un apprentissage par Q-learning, lorsque l'ensemble des états et/ou des actions possèdent un très grand nombre d'éléments, les dimensions de la Q-table deviennent proportionnellement plus grandes. Les demandes en mémoire et en capacité de calculs seront beaucoup trop élevées. Les limites du Q-learning mènent à l'apprentissage par Deep Q-learning (DQN).



Comparaison Q-learning et Deep Q Learning

Le deep learning (apprentissage profond) est un ensemble de méthodes du machine learning, qui s'appuie sur des réseaux de neurones artificiels, s'inspirant du monde vivant : le cerveau humain. Le réseau est composé de couches de neurones, dans lesquelles des données sont traitées en leur attribuant un poids. Plus un réseau possède de couches, plus il est possible de réaliser des tâches complexes.

Les jeux vidéo sont un exemple courant représentatif de l'utilisation du DQN. En effet, l'environnement que présente les jeux vidéo est large, chaque état de l'environnement peut être associé à un ensemble de pixels, dans lequel un agent peut choisir plusieurs actions. Dans ces conditions, il est effectivement difficile voire impossible de calculer toutes les actualisations de la Q-value pour toutes les combinaisons d'état et d'action.

C'est pourquoi en DQN, au lieu de calculer directement la Q-value et trouver l'optimale Q-function, nous utilisons une fonction d'approximation pour estimer l'optimale Q-function grâce à des réseaux de neurones profonds.

Le DQN répond au défi de l'entraînement par apprentissage renforcé : avoir des données d'entrées et d'une cible constamment changeante lors du processus, impliquant un entraînement instable. Ces données sont changeantes car elles sont corrélées, la mise à jour d'une variable engendre un impact sur les autres variables. Concrètement, lorsque la nouvelle Q-value est approximée plus grande, une nouvelle Q-value optimale est aussi approximée plus grande, provoquant l'effet d'un chien qui court après sa queue. La corrélation est tout simplement due à la mise à jour des poids du réseau lors du calcul d'une nouvelle Q-value, impliquant naturellement lors du calcul de l'optimale Q-value, une même tendance. De ce fait, le DQN apporte les nouveautés suivantes pour résoudre ce défi :

- Target network : créer deux réseaux dont l'un pour calculer les Q-values optimales (Target network) tandis que le second calcule les Q-values en incluant toutes les mises à jour des données notamment le poids du réseau, durant l'entraînement (Policy network). C'est seulement après un certain nombre de mises à jour de la policy network, que la target network va être à son tour mise à jour au poids de la policy network. Le but est de fixer temporairement la Q-value optimale afin de ne pas poursuivre une cible mouvante. Ceci permet de ne pas impacter directement la target network.

- Expérience replay : enregistre toutes les expériences passées et extrait un échantillon d'expériences aléatoires. L'aléatoire assure que les données soient moins dépendantes entre elles voire presque indépendantes et distribuées identiquement. En effet, les expériences arrivent séquentiellement, si le réseau apprend à partir de ces expériences consécutives, elles seront donc fortement corrélées.

Ces deux solutions nous permettent d'avoir des données d'entrées et de sorties plus stables pour entraîner le réseau. Ce sont donc ces méthodes que nous allons implémenter dans notre projet.

Définissons les 3 variables principales associées à notre DQN :

- State = (prix, demande, date), un état de dimension 3 et nous avons N possibles états.
- Action = indice qui se réfère à un état, et par conséquent un prix, une action est un entier parcourant la liste de tous les états possibles (varie de 0 à N).
- Environnement = récompensé par le nombre de nuits vendues * prix fixé – les coûts (représentant les frais airbnb soit les frais de ménage, de gaz et d'électricité).

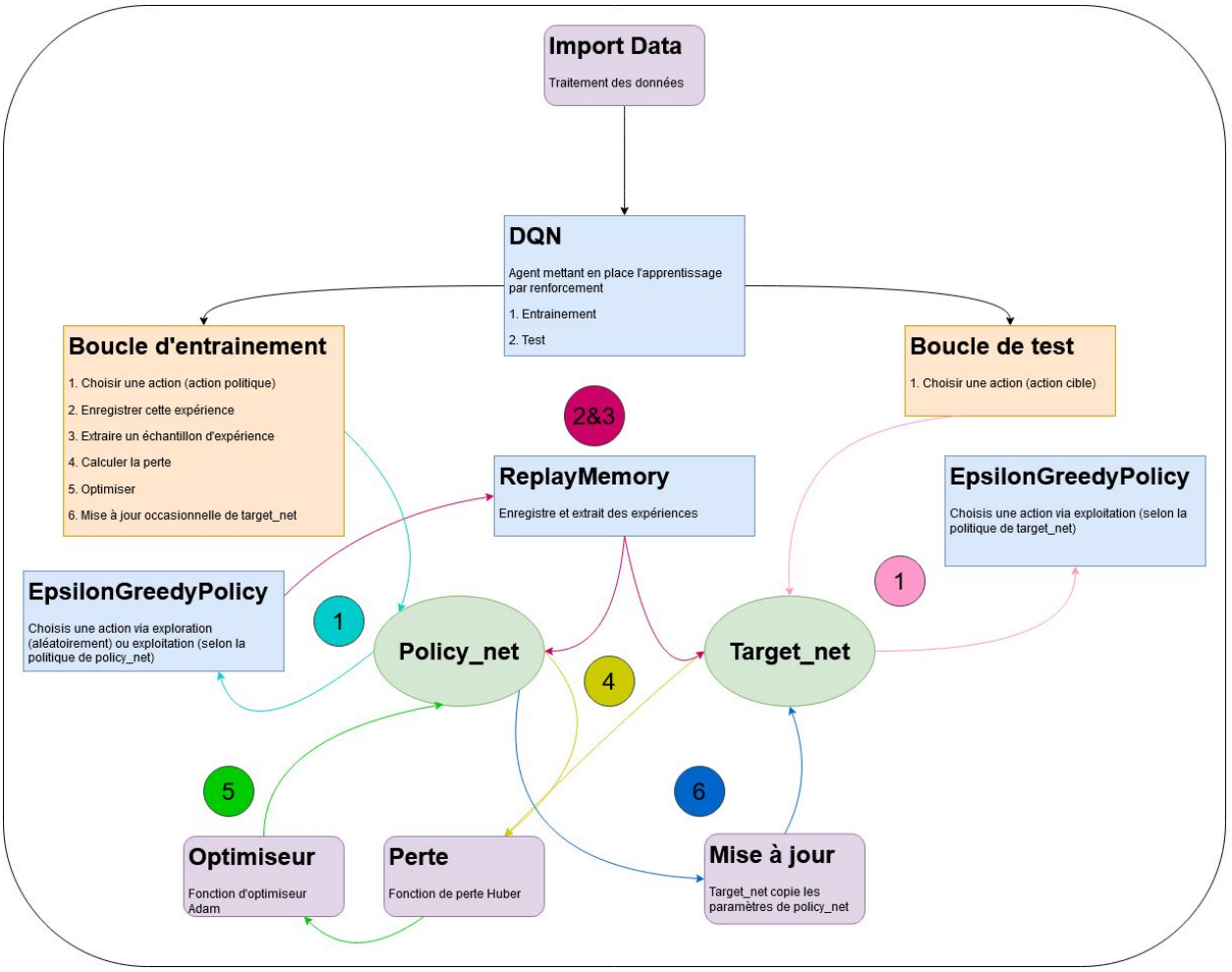


Diagramme des appels de classes côté vendeur (i.e. dans la classe DQN)

Nous avons donc créé une classe DeepQNetwork, permettant de former le réseau de neurones, de traiter les données reçues en entrée de celui-ci et de renvoyer en sortie les estimations de Q_Value. La fonction d'activation choisie est la ReLU (Rectified Linear Unit). Les fonctions d'activation usuelles en réseaux de neurones profonds sont la sigmoid et la tangente hyperbolique, mais ont été remplacées par la ReLU pour sa simplicité de calcul (lors du calcul du gradient, la dérivé est constante ou nulle pour des valeurs respectivement positives ou négatives) et donc sa rapidité d'exécution. C'est l'une des raisons principales pour laquelle nous avons choisi cette fonction d'activation. Cette classe sera utilisée dans la classe DQN, pour créer la policy network et la target network.

Lors de l'apprentissage, la classe EspilonGreedyPolicy nous permet de choisir une action via exploration ou exploitation des données, en étape 1 du diagramme, grâce à un seuil epsilon qui décroît exponentiellement au cours de l'apprentissage.

Après avoir réalisé une action, l'environnement se met à jour et nous recevons une récompense et prenons connaissance de l'état prochain. Cette transition est une expérience que nous allons enregistrer dans la mémoire de l'agent, lors de l'étape 2 du diagramme, grâce à la classe ReplayMemory, notamment caractérisée par une capacité limitée de

stockage. Lorsque la limite est atteinte et que l'on souhaite encore enregistrer de nouvelles expériences, alors la plus ancienne expérience est écrasée en faveur de la plus récente.

Dès que nous avons accumulé assez d'expériences, nous extrayons un échantillon comme base d'appui d'apprentissage, c'est l'étape 3 du diagramme.

Pour que notre estimation de la Q_value converge vers la Q_value optimale nous calculons la différence ("la distance") entre elles. Pour cela nous utilisons la fonction de perte de Huber, en étape 4 du diagramme, présentant l'avantage d'être moins sensible aux cas extraordinaires par rapport à la fonction MSEloss et dans certains cas évite l'explosion du gradient. Ce calcul de perte est ce que nous cherchons à réduire, par l'aide d'un optimiseur présent dans la librairie Torch, l'optimiseur Adam, que nous voyons à l'étape 5 du diagramme. Nous avons choisi cet optimiseur car il a été conçu spécifiquement pour l'entraînement en réseau de neurones profond et a montré un gain de temps considérable.

La target network copie les paramètres de la policy network mais ne le fait qu'après avoir laissé la policy network apprendre pendant une durée appelée dans notre code la target_update et représentée par l'étape 6 du diagramme. Cette variable, target_update représente le taux auquel la target network se met à jour (en nombre d'épisodes).

La partie test reprend la même étape 1 de choix d'action mais cette fois ci selon la target_net et sans choix aléatoire, en effet l'agent ne doit plus explorer ses actions possibles mais exploiter celles qu'il a jugé comme étant les mieux récompensées .

La classe DQN regroupe ainsi toutes ces étapes d'apprentissage et de test, ainsi que des fonctions mettant en forme les résultats obtenus.

B. Côté Clients

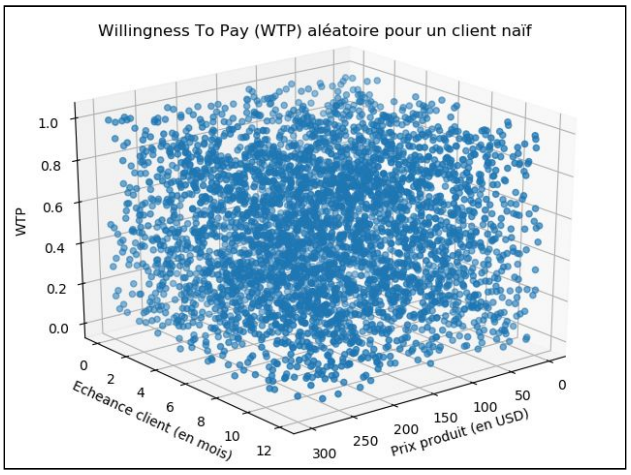
Pour estimer la volonté qu'un client veuille acheter un produit, on calcule le **Willingness-To-Pay (WTP)**. Le Willingness-To-Pay ou encore propension à payer est une probabilité entre 0 et 1 qui dépend du prix et du temps d'attente qu'il reste au client pour recevoir le produit (appelé ici échéance client, en mois). Plus la valeur du WTP est proche de 1, plus la probabilité d'achat est élevée. Nous verrons que ces approches d'estimations du WTP se sont faites différemment selon les types de clients. Dans la réalité, il est évident que le WTP ne dépend pas que de ces deux facteurs, mais afin de simplifier le modèle, ce sont ceux que nous avons choisi de conserver car ils sont aussi ceux que l'on peut quantifier le plus simplement.

1. Clients Naïfs

Les clients naïfs sont des clients qui représentent des personnes assez aisées qui ont des moyens conséquents et qui sont certains de partir. Ils ont donc en tête un prix minimum et un prix maximum et également une échéance. On fixe ensuite le WTP du client.

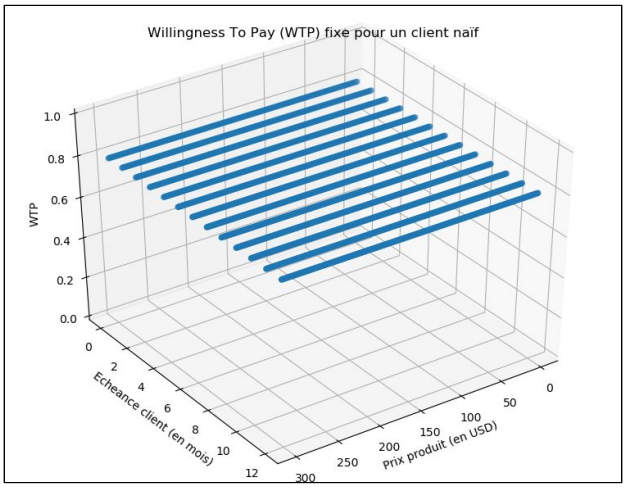
A ce moment là on va faire la distinction entre deux types de clients naïfs:

- Les premiers, avec un WTP fixé aléatoirement ce qui aboutit à une répartition uniforme comme présentée dans le graphe ci-dessous:



Willingness To Pay (WTP) aléatoire pour un client naïf

- Les seconds ont un WTP fixé élevé.



Willingness To Pay (WTP) fixe pour un client naïf

L'objectif de cette distinction est d'assurer un certain pourcentage des ventes réelles observées dans les données. En faisant cela, nous essayons de conserver les variations cycliques de ces données. Lors du développement de notre projet nous avons utilisé une proportion de 15% de clients naïfs avec un WTP fixé à 0,85.

Nous avons mis en place un comportement d'achat instantané dans les cas où le prix proposé soit inférieur au prix minimum attendu par le client, nous pourrions comparer cela à une réaction face à une promotion.

Dans le cas où le prix est supérieur au prix maximum attendu, le client attend simplement le mois suivant pour espérer avoir un prix inférieur. Il passe à l'achat quand le prix est au-dessus de son prix maximum seulement lorsque l'échéance est à 0 et qu'il reste de la place sur le mois désiré.

2. Clients Stratégiques

Le client stratégique se distingue du client naïf par sa capacité d'analyse et de décision. Il analyse le marché, notamment ses prix historiques afin d'en tirer des conclusions sur lesquelles sera basée sa décision d'achat.

Il convient tout d'abord de déterminer une fourchette de prix acceptables.

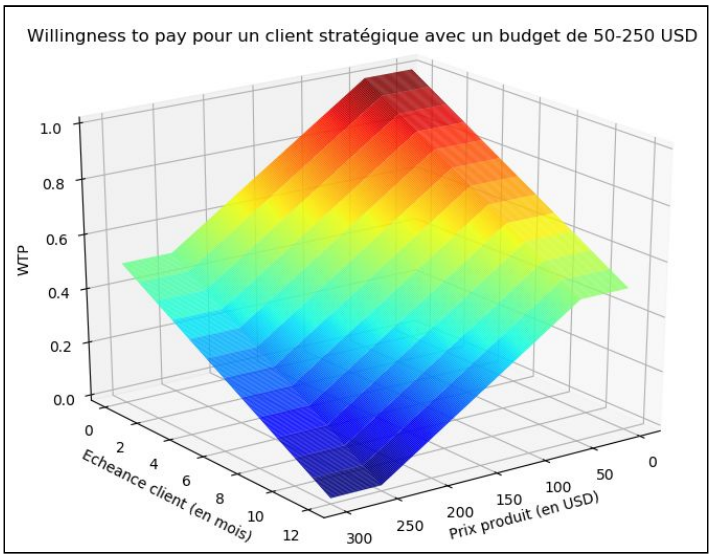
Le calcul des valeurs minimales et maximales d'estimation de prix par un client stratégique prend en compte les données de prix historiques du produit.

La moyenne des 20% des prix les moins élevés constitue la fourchette basse de prix acceptables par le client, ceci évite les écarts trop importants des derniers prix, et permet de se placer intelligemment dans la fourchette globale de prix. Pour le prix maximal, nous utilisons une solution équivalente, prenant cette fois-ci les 20% des prix les plus élevés, pour les mêmes raisons que celles citées précédemment.

Dans le cas du client stratégique, le Willingness-To-Pay (WTP) est calculé à partir des données historiques, prenant en compte le prix des réservations et l'échéance client, tandis que, rappelons-le, du côté du client naïf, son WTP est soit fixé, soit totalement aléatoire.

Concernant le calcul du WTP brut du client stratégique: il évolue de façon linéaire en fonction du prix du produit et de l'échéance client. L'échéance client est comprise entre 0 et 12 mois, elle représente le nombre de mois restant avant son départ envisagé, sa valeur est ainsi comprise entre 0 et 12 en valeurs entières. Quant aux prix, cela est conditionné par les prix minimum et maximum que le client se permet de dépenser, expliqués ci-dessus. Ainsi, si le prix du produit est supérieur à son prix maximum, la réservation est repoussée au mois suivant, et au contraire si le prix est inférieur à son prix minimum qu'il est prêt à débourser, la réservation est immédiate, tout comme le fait un client naïf.

D'une manière générale, plus le prix est bas, plus le client stratégique aura de chances de réserver. De même, plus l'échéance est basse, plus les chances de réservation sont élevées. Ces deux paramètres ont un poids égal dans le calcul du WTP, ce qui permet à la fin d'obtenir le WTP brut du client stratégique dont vous trouverez ci-dessous une représentation graphique. Elle est comprise entre 0 et 1, le 0 signifiant 0% de volonté d'achat, et 1 pourrait représenter une volonté d'achat infaillible.



Willingness To Pay (WTP) pour un client stratégique

De plus, nous avons ajouté une couche supplémentaire de réflexion : le client stratégique accordera davantage d'importance à l'échéance. Lors de la prise de décision du client, le poids du paramètre de l'échéance sera bien plus important que le prix en lui-même, mais reste tout de même compris dans sa pensée. Nous avons ainsi ajouté une variable de conditionnement de la réservation, et pour la calculer, sont pris en compte les paramètres de l'échéance, le prix et un paramètre aléatoire pour être plus proche des comportements variants de la réalité.

C'est en cela que cette variable de conditionnement est différente du WTP brut. Cette variable fonctionne presque comme le WTP, mais avec un paramètre supplémentaire comme cité plus haut : l'ajout de l'aléatoire. De la même manière, plus le prix est bas, plus cette variable de conditionnement sera élevée, et aussi, plus l'échéance est basse, plus elle sera élevée également. La valeur de cette variable est elle aussi comprise entre 0 et 1. Tout en prenant en compte que le paramètre de l'échéance a un poids très supérieur dans le calcul, et les paramètres du prix et de l'aléatoire ont chacun un poids inférieur à celui de l'échéance. Ces paramètres restent modulables.

Le Willingness-To-Pay de chaque client sera différent, même pour une échéance client et un prix de réservation identiques, car chaque WTP est adapté à chaque client. En effet, le WTP varie plus ou moins fortement en fonction des prix minimum et maximum que le client est prêt à débourser. Le paramètre du prix prend en compte ces bornes du client et ainsi attribue un WTP correspondant à chaque situation, ce qui est plus représentatif du monde réel.

De plus, nous avons rajouté une autre condition pour que le client stratégique procède à la réservation. En général, ces clients stratégiques, n'achètent pas directement dès le premier prix qu'ils rencontrent, mais préfèrent le comparer aux autres, pour tirer le meilleur profit possible et faire la meilleure acquisition en terme de prix. Ainsi ici, un client stratégique comparera le prix qu'il rencontre à l'instant T aux deux prix rencontrés précédemment. Si ce prix est inférieur ou égal à ces deux derniers, il va aboutir à la réservation. Dans le cas contraire, il va attendre et repousser son achat jusqu'à, peut-être remplir cette condition plus tard. Cela lui permet de toujours réserver pour le meilleur prix possible, qui lui est primordial.

Tous ces paramètres de réflexion, permettent au client stratégique de se démarquer du simple client naïf, et le rendent ainsi “stratégique”.

C. Le marché

Le point clé de ces deux parties précédentes est l'endroit où ils se rencontrent : le marché. Nous l’avons modélisé comme le présente le schéma ci-dessous :

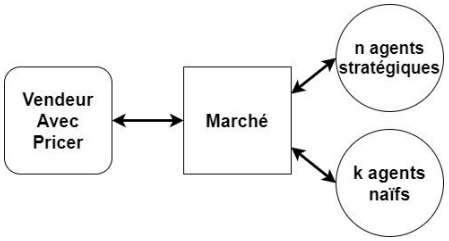


Schéma d'interactions entre les parties

En pratique, nous réalisons des tests sur une durée théorique de 12 mois de mise en location ce qui veut dire que cette interaction est répétée 12 fois pour avoir un cycle complet puis nous effectuons ce cycle autant de fois qu’il y a de proportions différentes entre les clients stratégiques et les clients naïfs.

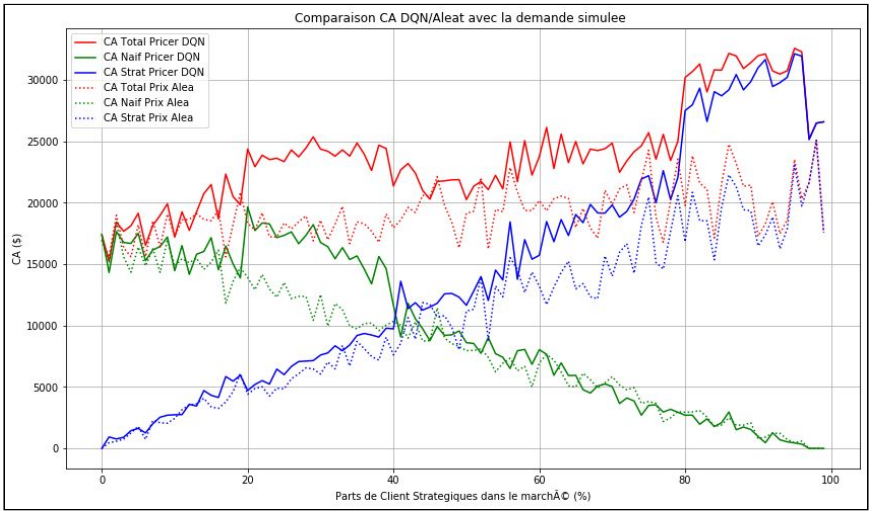
De plus, nous avons fait le choix de conserver un niveau constant de clients chaque mois, c’est-à-dire que le nombre de clients entrant était équivalent au nombre de clients sortant, tout en conservant les proportions comportement inchangées évidemment. Au fur et à mesure des avancées, les échéances du dernier mois étaient donc logiquement plus nombreuses.

VI. Tests et Réalisation

L'idée initiale est de louer un appartement dont les premières estimations sont d'un prix de 150$ la nuit. Nous testions alors le DQN sur une proportion bien spécifique de clients stratégiques et naïfs dans le marché. Nous comparons notre DQN à des prix aléatoires sélectionnés dans un intervalle de ±20$ du prix initialement fixé afin de "simuler" les variations manuelles qu'aurait pu fixer un propriétaire souhaitant voir ce qu'engendrerait une variation sur ses ventes. Les prix du vendeur témoin sont fixés chaque mois pour chaque proportion.

Nous avons alors souhaité entraîner notre DQN une fois sur toutes les proportions de clients stratégiques / naïfs existant. En toute franchise, les résultats attendus n'étaient pas au rendez-vous. Nous avons donc décidé de nous rapprocher des hypothèses que pourrait faire un utilisateur en appliquant le DQN à des proportions spécifiques du rapport stratégiques/naïfs. C'est-à-dire que nous entraînons un DQN sur une proportion de clients stratégiques allant par exemple de 0 à 20% des clients totaux du marché.

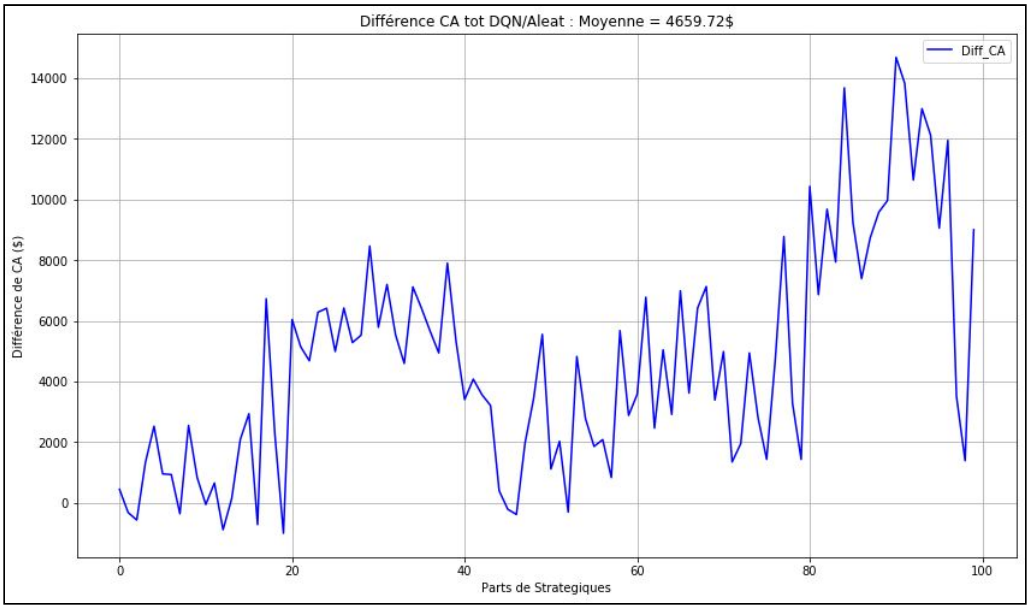
Cette petite modification a tout de suite amélioré nos résultats. En effet, le DQN a pu s'adapter de façon beaucoup plus spécifique aux différences de comportement en fonction des proportions. Avec 5 DQN entraînés sur chaque 20% des proportions voici ce que nous avons obtenu.



Graphique de Comparaison du chiffre d'affaires issus des prix DQN et du vendeur témoin

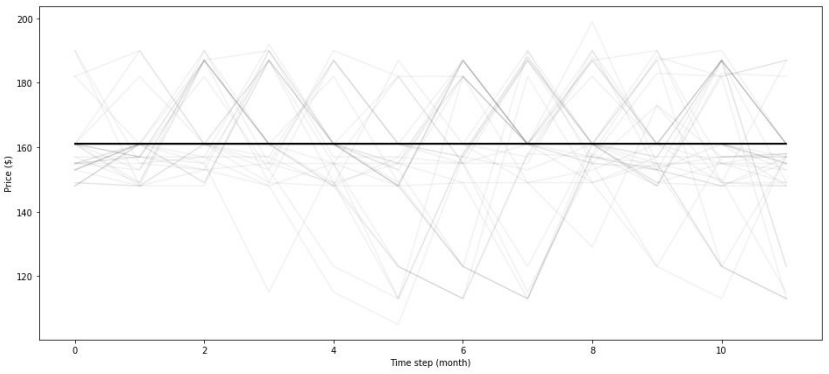
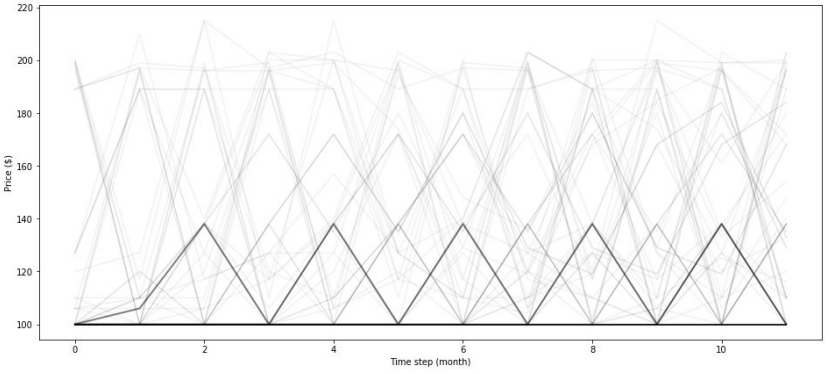
Tout d'abord une explication quant à la lecture du graphique: prenons l'exemple de 80% de clients stratégiques sur le marché. Le chiffre d'affaires total (Prix de vente x nombre de ventes) issu des prix fixés par le DQN est de 31317$, sur cette somme les clients stratégiques participent à hauteur de 27646$ et les clients naïfs à hauteur de 3671$. Nous comparons alors cela au prix aléatoire fixé par notre vendeur témoin.

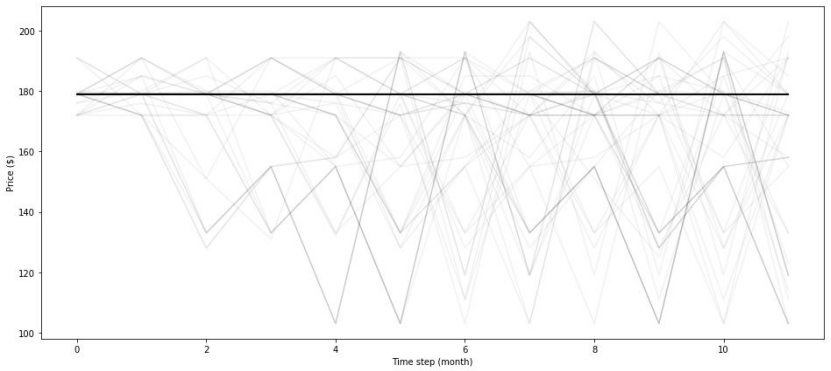
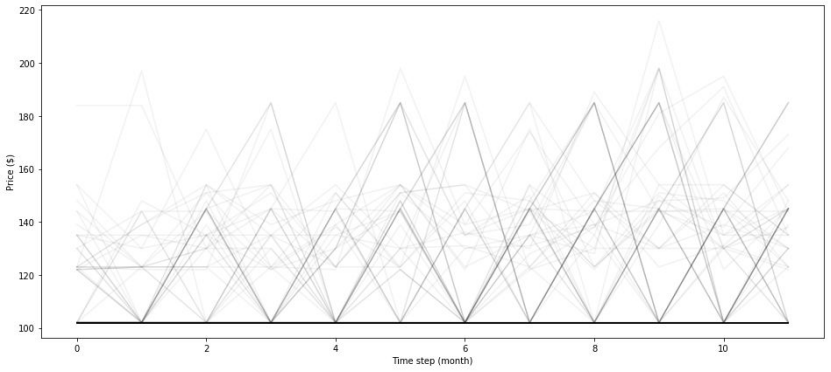
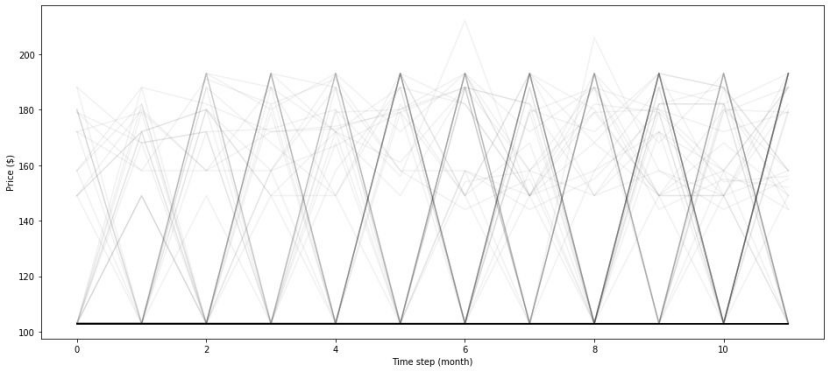
Comme nous pouvons le voir, le DQN est très performant lorsque le marché est composé de plus de 80% de clients stratégiques. Et cela se voit très bien dans le graphique ci-dessous qui montre la différence entre le CA réalisé par le DQN et le CA témoin.



Graphique de d'écart de chiffre d'affaires entre le DQN et le vendeur témoin

La moyenne de différence de CA observée sur toutes les proportions est de 3140$. Cette moyenne positive montre que le DQN apporte un avantage significatif au vendeur qui en bénéficie. Ce qui nous a alors étonné fut les prix appris par le DQN sur les différentes proportions par rapport aux prix réellement appliqués lors de la mise en situation :

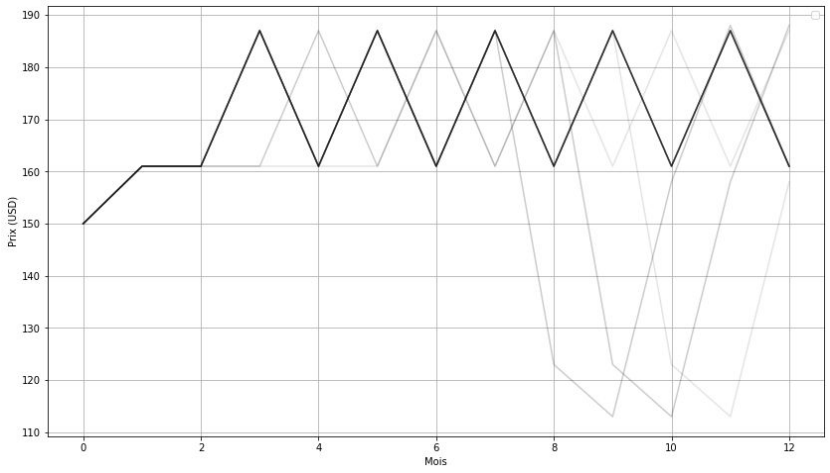
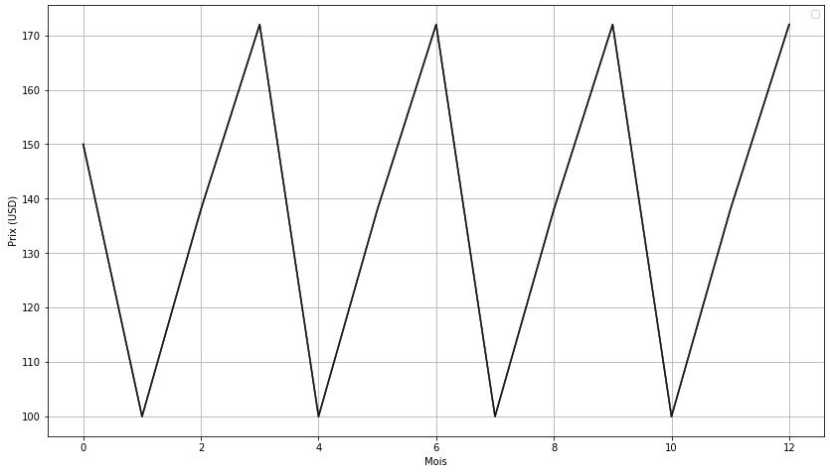
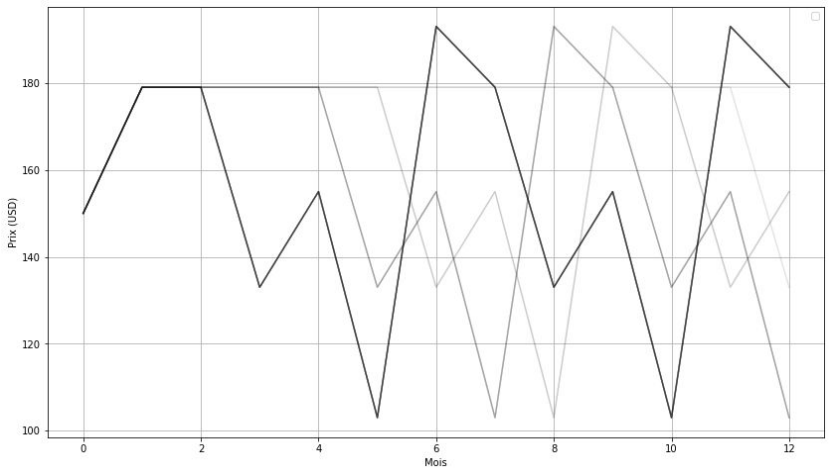
0% - 20% Clients Strat.	
20% - 40% Clients Strat.	

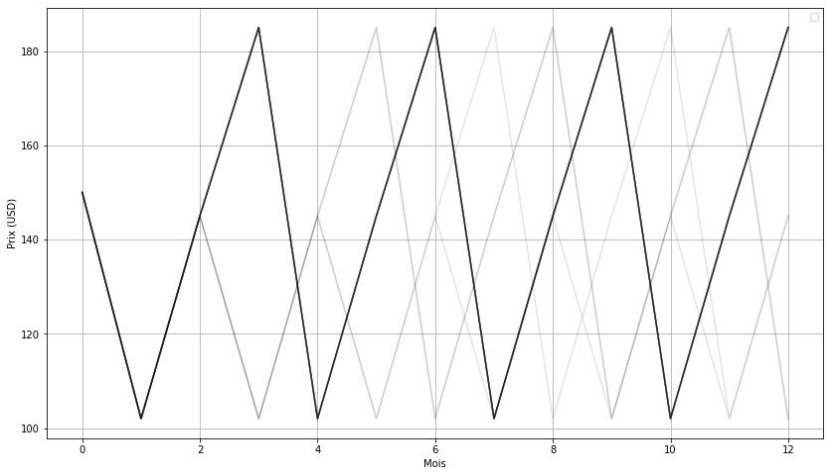
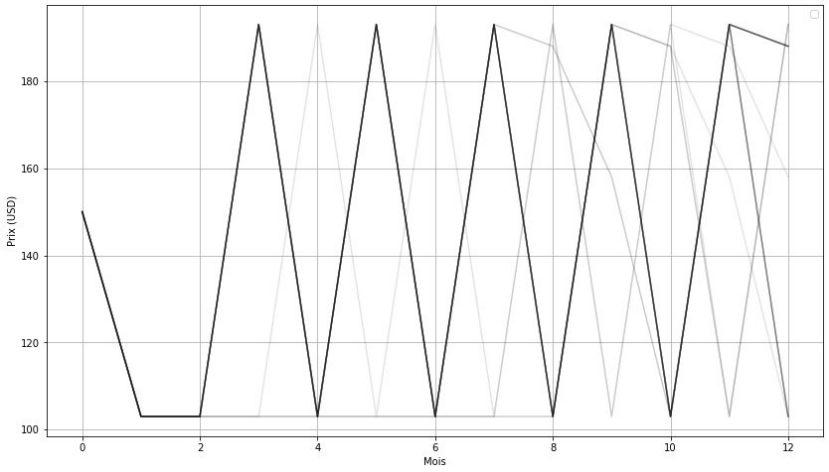
40% - 60% Clients Strat.	
60% - 80% Clients Strat.	
80% - 100% Clients Strat.	

Ces graphiques représentent donc les séquences de prix envisagées lors des différents épisodes de test (ici 500 par portions). On peut voir que le DQN s'entraîne majoritairement sur des prix fixes mais explore tout de même d'autres possibilités de variations. C'est cet équilibre entre exploration et exploitation qu'il est parfois difficile à trouver.

Nous pouvons noter que les prix fixés explorés sont assez élevés, ou en tout cas supérieurs à l'estimation initiale de 150$, lorsque les clients naïfs sont majoritaires et qu'ils sont plus bas lorsque les clients stratégiques sont majoritaires.

Voyons maintenant les prix réellement appliqués lors de la mise en situation :

0% - 20% Clients Strat.	
20% - 40% Clients Strat.	
40% - 60% Clients Strat.	

60% - 80% Clients Strat.	
80% - 100% Clients Strat.	

Ce que nous observons est la superposition des 20 séquences de prix de chaque DQN. On notera les très importantes variations de prix dès que le nombre de clients stratégiques est non négligeable (>20%) et la séquence de prix inédite sur la portions de transition 40 - 60% avec un cycle comportant 2 hausses et 2 baisses.

Nous remarquons également que sur certaines suites de prix le DQN présente un décalage des cycles par rapport à la majorité, cependant son *pricing* reste sensiblement similaire dans les niveaux de prix fixés.

VII. Interprétations et Difficultés

Interprétations

Ces résultats sont satisfaisants puisque notre DQN est meilleur que notre vendeur témoin qui représente un propriétaire qui ferait varier ses prix aléatoirement pour voir comment cela influerait sont CA. De plus, notre *pricer* sort largement gagnant lorsque les clients stratégiques sont largement majoritaires ce qui est assez encourageant quant à la pertinence de l'utilisation du DQN.

Remarquons que sur la partie où les clients naïfs sont majoritaires les bornes de prix dans lesquelles le DQN varie sont élevées car il exploite le caractère obligatoire du départ des clients naïfs. En effet, rappelons-le, si ceux-ci n'ont pas réservé d'appartement quand leurs échéances arrivent à zéro, s' il reste des disponibilités, ils paieront le prix fixé quel qu'il soit pour partir.

La seule portion qui reste relativement décevante est celle comprise entre 40 et 60% de clients stratégiques. C'est également la seule où le DQN met en place un cycle comprenant une diminution discontinue. Nous pouvons attribuer ce comportement au fait qu'il ne sait pas comment se positionner entre exploiter le comportement des clients naïf ou celui des clients stratégiques.

De façon plus globale, nous observons que le DQN finit toujours sa séquence sur un prix élevé. Effectivement, comme le nombre de clients est constant, le nombre de ceux qui ont une échéance au dernier mois s'accumule. Le DQN exploite donc cette modélisation afin de vendre au plus cher lorsque le maximum de clients ont les WTP les plus élevés.

Enfin, il semblerait que le CA global général semble augmenter lorsque que la part de clients stratégiques augmente dans le marché; et cela même si les prix que le DQN pratique sont extrêmement bas.

Difficultés

Les difficultés ont été nombreuses sur ce projet, mais nous avons réussi à les surmonter et à s'adapter à chaque fois que nous y faisions face ce qui à été extrêmement formateur.

Commençons par les données, initialement nous souhaitions *pricer* les billets de concerts ou d'évènement sportifs. Après 2 mois de recherche n'ayant toujours rien trouvé, nous avons bifurqué sur les données d'AirBNB afin d'avoir le mois de rush entier pour se consacrer au développement du code de notre projet. Il semblerait que les données de prix des évènements soient extrêmement précieuses car avec la vente et la revente sur internet la concurrence est de plus en plus rude.

Ensuite, voyons le développement du DQN qui était certainement le point le plus dur techniquement de notre réalisation. En effet, il a fallu comprendre profondément chaque concept afin de restituer un code qui l'exploite au maximum et en tire le meilleur. De plus, la capacité de calculs de chaque ordinateur étant différente, il a fallu adapter nos périodes de test à celles-ci. Pour nos derniers tests qui sont les plus exigeants en puissance nous avons dû les effectuer sur un poste fixe.

Puis, déclinons l'interprétation des résultats qui fut la source de beaucoup de doutes et d'énormément de remises en question. Nous avons alors passé en revue notre modélisation des clients, du marché et notre DQN de A à Z. Nous avons ensuite essayé de chercher des paramètres plus optimaux. Mais maintenant nous sommes sûrs de nos résultats et convaincus qu'ils sont corrects par rapport à notre modélisation du marché.

Nous pouvons également aborder les difficultés qu'implique la situation que nous traversons car en temps qu'étudiant, bien qu'organisant notre temps comme nous le souhaitions, il est souvent difficile de faire face aux imprévus, aux obligations, aux baisses de moral et aux doutes de chacun. Nous avons donc dû rester flexibles et faire preuve d'écoute et de compréhension afin de rester bienveillants les uns envers les autres.

Comme nous l'avons dit précédemment, ce sont ces difficultés qui ont été formatrices lors de ce projet, qui nous ont soudé et qui nous ont permis d'avancer.

VIII. Impact (RSE)

L'enjeu environnemental est de plus en plus important et pris en compte dans la conception d'un projet. Il faut avoir une conception durable du produit final.Nous allons décrire l'impact social et environnemental de notre produit.

Tout d'abord concernant l'impact social. Notre projet permet donc de créer un bénéfice plus important ce qui enrichit donc les propriétaires et augmente inévitablement, sur le long terme, leur niveau de vie. On peut aussi penser en tant qu'investisseur, et se dire que pour encore augmenter sa marge, on pourrait essayer de faire diminuer les charges d'électricité en assurant un bon isolement du bien et des installations électriques avec des niveaux de consommations les plus bas possible.

Plus généralement, pour des utilisations de notre DQN autres que sur des locations AirBNB, l'optimisation des prix pourrait permettre aux vendeurs d'être plus rapidement rentables. Ceci est un aspect à ne pas négliger puisque dans la situation actuelle il est important pour les commerçants qui font une transition digitale de faire le maximum de chiffre d'affaires au plus vite afin d'essayer de combler au plus vite les pertes engendrées par les précédents confinements.

De plus une utilisation efficace sur des produits de consommation permettrait d'écouler des stocks plus efficacement, évitant la surproduction en cas de sous-vente et permettrait de prévoir une production en vue de ventes importantes à venir.

Les impacts négatifs du projet sont donc assez minimes à partir du moment où l'électricité utilisée pour alimenter le support qui exécute l'algorithme est produite de façon durable. L'optimisation est donc également un sujet important à prendre en compte lors du développement, mais pour cette version nous avons principalement priorisé le résultat.

IX. Perspectives

Nous sommes heureux d'avoir travaillé à la réalisation de ce projet même s'il reste encore des améliorations à apporter pour arriver à un produit final pleinement exploitable. Nous allons vous énumérer les améliorations que nous pensons être pertinentes pour ce projet.

Tout d'abord, améliorer le DQN pour qu'il soit plus rapide d'exécution et le plus précis possible en utilisant des optimiseurs de paramètres car, comme ils sont très nombreux dans un DQN, il est important de chercher l'optimalité pour tous afin d'avoir un équilibre exécution / précision cohérent.

Nous pouvons également penser à l'amélioration du réseau de neurones en introduisant plus de couches. En effet, plus le nombre de couches est important dans le réseau, plus il apprend, s'entraîne pour donner en sortie un résultat plus précis.

Nous pensons également à l'intégration d'autres facteurs comme la surface, la localisation ou encore le nombre de lits d'un appartement. Le choix de ces critères est propre à chaque type de produit, donc avant d'améliorer cette partie, il faudra essayer de rendre plus dynamique cette sélection à travers une couche graphique par exemple.

Ensuite, nous avons pensé à quelques améliorations du côté du modèle des comportements clients.

Par rapport aux clients stratégiques, nous pouvons complexifier les critères d'achats pour rendre le comportement le plus réaliste possible. En dehors du prix, nous pouvons ajouter la qualité d'un produit. On peut aussi penser à un modèle de client intelligent qui serait contrôlé par une intelligence artificielle.

Concernant le marché on peut également penser à créer un marché concurrentiel en exposant plusieurs prix de différents vendeurs aux clients. Ceci permettrait de se rapprocher encore un peu plus de la réalité.

En revenant au problème d'AirBNB, dans un contexte plus global des problématiques d'un propriétaire, la première estimation du prix du bien est un point clé de la location AirBNB. Compte tenu de la quantité de facteurs influant sur celui-ci, il semblerait que le problème d'estimation du premier prix soit un problème de clusterisation. Il sera alors intéressant, dans un contexte global de ce problème de commencer par une estimation initiale par clusterisation puis d'appliquer un DQN pour les faire varier dynamiquement.

X. Conclusion

Travailler sur ce projet a été très instructif. En effet, nous avons élaboré un projet qui est adaptable pour des produits de la vie de tous les jours et pour tous les commerces en ligne.

D'un point de vue pédagogique, nous avons pu approfondir nos connaissances en Machine Learning ainsi qu'en programmation Python. Le projet demande des compétences informatiques en Machine Learning assez spécifiques et nous avons dû apprendre à manipuler des librairies d'apprentissage par renforcement comme PyTorch.

De plus, nous avons appris à travailler ensemble sur un sujet complexe, à essayer de le décomposer ensemble et à utiliser les compétences et les envies de chacun pour le mener à bien. Nous avons aussi compris l'importance de faire des réunions régulières d'avancement de projet afin de réajuster les actions de chacun si nécessaire au fur et à mesure.

L'approche globale de la modélisation des interactions du marché a été un véritable plus. Plus particulièrement le fait que nous ayons divisé le travail au sein de l'équipe en sous groupe fut très formateur au moment des mises en commun car nous devions réellement comprendre ce que faisaient les autres afin que les sous-programmes s'imbriquent correctement.

Enfin, les parties les plus instructives du projet furent les difficultés rencontrées, aussi bien au niveau de la recherche initiale des données, qu'à la mise en place du DQN et des clients pour finir par l'interprétation des résultats. Le projet à été un enchaînement de petites difficultés auxquelles nous avons dû faire face sans jamais abandonner et nous pensons que c'est cela le plus formateur : ne pas baisser les bras. En effet, conclure notre scolarité à l'ECE en réalisant une projet qui montre que le travail paye même si les résultats semblent étonnants fait paradoxalement écho au slogan de l'école : *"là où tout devient possible"*.

Nous sortons donc grandi de cette expérience, de ce projet et des connaissances que nous avons pu acquérir lors de sa réalisation.

XI. Références Bibliographiques

A. Articles de recherche

1. Apprentissage par renforcement et DQN côté vendeur

[1] R. Maestre, J. Duque, A. Rubio, and J. Arevalo, “Reinforcement Learning for Fair Dynamic Pricing”, [Online]. Intelligent Systems Conference 2018, 6-7 September 2018, London, UK https://arxiv.org/pdf/1803.09967.pdf.

[2] Fei Huang and Hao Huang, Event Ticket Price Prediction with DeepNeural Network, on Spatial-Temporal Sparse Data. ACM, New York, NY, USA, Article 4, 8 pages. 2020 https://arxiv.org/pdf/1912.01139.pdf

[3] Lennard Alexander Kropp, Jakob J. Korbel, Max-Marcel Theilig, Rüdiger Zarnekow, Dynamic Pricing Of Product Clusters: A Multi-agent Reinforcement Learning Approach, Twenty-Seventh European Conference on Information Systems (ECIS 2019), Stockholm-Uppsala, Sweden, 2019 https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1031&context=ecis2019_rp

[4] Fenjiro Y., Benbrahim H, Deep reinforcement learning overview of the state of the art, 2018 http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-c799f70c-d6bb-4e75-9bba-ef2313881015

[5] Rupal Rana, Fernando Oliveira, Real-Time Dynamic Pricing in a Non-Stationary Environment using Model-Free Reinforcement Learning, 2014 https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi58KOqotXsAhUFgRoKHaz2Cv8QFjABegQIAhAC&url=http%3A%2F%2Fwww45.essec.edu%2FprofessorsCV%2FshowDeclFileRes.do%3FdeclId%3D11396%26key%3DPublication-Content&usg=AOvVaw1YcUN7ExUBnLiKP-TdgE-u

[6] Deep Reinforcement Learning Algorithm for Dynamic Pricing of Express Lanes with Multiple Access Locations, Venktesh Pandey, Evana Wang, and Stephen D. Boyles, 10 sept 2019
Lien article : https://arxiv.org/pdf/1909.04760.pdf
Source code github : https://github.com/venktesh22/ExpressLanes_Deep-RL

2. Effets et modélisation du comportement clients

[7] Pricing Decisions for a Sustainable Supply Chain in the Presence of Potential Strategic Customers, Xinmin Liu, Kangkang Lin, Lei Wang and Lili Ding, 22 February 2020 https://www.mdpi.com/2071-1050/12/4/1655/htm

[8] Su, X. Intertemporal Pricing With Strategic Customer Behavior. Management Science, 53(5), 726-741, 2007 https://repository.upenn.edu/cgi/viewcontent.cgi?article=1155&context=oid_papers

[9] Rainer Schlosser, Stochastic Dynamic Pricing with Strategic Customers and Reference Price Effects, February 2019
https://www.researchgate.net/publication/331375444_Stochastic_Dynamic_Pricing_with_Strategic_Customers_and_Reference_Price_Effects

[10] Yossi Aviv, Mike Mingcheng Wei, Fuqiang Zhang, Responsive Pricing of Fashion Products: The Effects of Demand Learning and Strategic Consumer Behavior, 2019
https://pubsonline.informs.org/doi/10.1287/mnsc.2018.3114

[11] Yang Zhao, Guojun Ji, Yue Jiang and Xiaopei Dai, Strategic Customer Behavior and Pricing Strategy Based on the Horizontal Differentiation of Products, 18 Jan 2020
https://www.hindawi.com/journals/mpe/2020/9475150/

[12] Multi-agent Simulation for the Manufacturer's Decision Making in Sharing Markets, Hiroki Takahashi, Nariaki Nishino, Takeshi Takenaka, Available online 21 March 2018
https://reader.elsevier.com/reader/sd/pii/S2212827117312040?token=CB6DC88A919C75F7F39631FCF5EBC29C55B1B84A021E9375E70CDCB6F80775E397C89040C011BAA9C9504309D2EB7A3F

B. Etat de l'art

Net Rivals: https://www.netrivals.com/fr/
Omnia Retail: https://www.omniaretail.com/
Price2SPy: https://www.price2spy.com/

C. Aide à la compréhension et à la programmation

Optimiseur Adam
https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

Fonction d'activation ReLU
https://deepai.org/machine-learning-glossary-and-terms/relu
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Tutorial pytorch
https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

Blog sur l'optimisation de prix et supply chain
https://blog.griddynamics.com/deep-reinforcement-learning-for-supply-chain-and-price-optimization/?fbclid=IwAR1_RoAHBy9sCfUFUM-L35oydZIkQ2iPnjcwazAB6dFgJElzwWXniu58CUw

Tutorial reinforcement learning
https://deeplizard.com/learn/video/PyQNfsGUnQA

Github
https://github.com/xkiwilabs/DQN-using-PyTorch-and-ML-Agents

D. Données Airbnb

http://insideairbnb.com/