Projet de data-mining - 2020/21

Emna BARRED
Nabil LAMRABET
Guillaume ORTEGA
Hedwin BONNAVAUD

Les différents codes sources et jeux de données peuvent être récupérés depuis le dépôt github suivant : https://github.com/nalmt/datamining/tree/main.

Introduction

En octobre 2020 le marché du jeu vidéo pesait plus de 159 milliards de dollars, dépassant le poids de celui de la musique. Une industrie où depuis 40 ans joueuses et joueurs achètent et jouent à des jeux vidéos quotidiennement. Le savoir-faire français reconnu internationalement se positionne tous les ans parmi les cinq premiers vendeurs de jeux vidéo à travers le monde, pour la France le jeu vidéo représente la première source de PIB en exportation culturelle. Un marché qui reste encore à explorer et où genre et tendances se sont déjà installés. De grands noms ont déjà fait leur place mais les plus petits ne sont pas en reste, le jeu vidéo indépendant étant de plus en plus en vogue sur le marché du jeu dématérialisé.

Genre et tendances donc, mais lesquelles ? Si les jeux vidéo s'articulent en block-buster ou en tendances à l'image du cinéma, quel est le studio qui a instauré quel code ? Quelle place pour une nouvelle production ? Est-ce que la sortie d'un jeu peut-être à l'avance catégorisée comme un futur succès ou est ce que le marché du jeu ne répond finalement à aucun code, et les ventes sujettes aux humeurs des acheteurs ?

Voilà les questions que nous nous sommes posées et nous allons tenter de répondre à la plupart d'entre elles. Pour cela nous nous sommes focalisés sur le marché du jeu vidéo dématérialisé mais pas uniquement.

Nous avons donc trouvé intéressant d'utiliser plusieurs dataset comme ressource d'un logiciel capable de conseiller des joueurs dans l'achat de leurs jeux ou des développeurs dans les choix de leur production.

Nous décidons donc d'articuler notre projet en deux phases, la première étant le conseil d'achats de jeux pour un joueur donné en fonction de son activité vidéo-ludique passée, et la seconde le conseil aux studios de développement en fonction de l'état du marché.

Nous terminerons enfin ce rapport par une conclusion qui reprendra les questions éthiques que ce sujet impose.

Conseil d’achats de jeux pour les joueurs

Définition de profil de joueurs

On ne s’attardera pas sur le détail des traitements effectués, ceux-ci sont disponibles dans les différents Jupyter notebooks.

On souhaite déterminer le profil de nos joueurs, savoir s’il existe des groupes conséquents de joueurs avec similitudes sur leurs choix de jeux et enfin pouvoir leur recommander des jeux (qu’ils n’ont pas encore acheté) adaptés à leurs choix.

La première étape consiste à mettre le doigt sur les catégories préférées des joueurs. Pour cela on se sert de deux métriques :

- Les catégories des jeux achetés.
- Mais également du temps passé sur ces jeux.

On peut se demander si les catégories des jeux achetés suffisent. Simplement, un jeu est acheté car il correspond aux critères du joueur, même s’il n’est pas joué. Néanmoins il se peut que d’autres critères peuvent amener à l’achat d’un jeu (par exemple : l’achat résulte uniquement des soldes ou d’un prix attractif ; par exemple il existe beaucoup de jeux à moins d’un euro). De ce fait on décide d’ajouter le temps passé (telle une pondération), cette métrique est pertinente pour mesurer l’attrait d’une catégorie pour le joueur.

Nous utilisons les datasets de Steam contenant des données sur les joueurs (un Id par joueur, le nom du jeu du acheté (s’il a été acheté), le nombre d’heures jouées (uniquement si le jeu a été acheté et joué).

- Nous avons nettoyé le dataset (suppression de la colonne 0, suppression des doublons).
- Nous avons modifié le dataset pour n’afficher que les heures jouées. Les jeux non joués ont 0 heures de jeu. Néanmoins nous avons décidé de leur donner 0.5 heures de jeu. 0 heures de jeu signifie généralement que le joueur a acheté le jeu mais n’est pas intéressé. Comme évoqué précédemment cela peut arriver lorsque le jeu coûte très peu cher ou en solde. Néanmoins, il se peut également que le joueur ait tout juste acheté le jeu au moment de la constitution de la base de données ou alors qu’il attend de finir un autre jeu avant. C’est donc pour contrebalancer ce biais qu’on décide d’attribuer arbitrairement 0.5 heures aux jeux non joués. Cette modification fut très discutée. Elle n’aura finalement presque aucun impact.
- On effectue un ensemble de traitements sur le jeu de données afin d’obtenir des catégories sous format “one hot” pour chaque jeu joué par un joueur.
- On multiplie les catégories du jeu par le temps passé sur le jeu (pour un joueur). L’objectif est de pondérer les jeux achetés par le temps de jeu.
- On agrège les données par joueur en effectuant une moyenne par catégorie. Ainsi on obtient des scores de catégories par joueur.

Nous avons effectué les mêmes traitements en adoptant le point de vue des Tags de jeu (à la place des catégories).

Les catégories (Valve Anti-Cheat enabled , Steam Achievements, Cross-Platform Multiplayer, Steam Trading Cards, Single-player, Multi-player etc.) sont des informations globales là où les Tags sont plus spécifiques quant à la nature du jeu (Wrestling, Zombies, World War 2, E-sport etc.).

Clusterisation des joueurs selon leurs catégories préférées

Désormais on souhaite connaître s'il existe des groupes de joueurs ayant des goûts similaires et si certains de ces groupes sont de taille importante. En effet, si on découvre des groupes de joueurs aux goûts similaires importants, il sera plus intéressant de leur proposer des recommandations de jeu.

Nous avons donc effectué quelques traitements :

- Normalisation des données
- Analyse en composantes principales, réduction du nombre de variables (en conservant au moins 80% des données), sélection du nombre de composantes.
- Calcul de l'erreur quadratique pour l'algorithme K-means et recherche du nombre de clusters idéal.
- Affichage des données sous forme de nuage de points en 3 dimensions.

Nous avons essayé quelques algorithmes de clusterisation avec différents paramètres (GaussianMixture, SpectralClustering, DBSCAN, Birch et K-means). Birch et K-means nous offrent un résultat similaire et une clusterisation qui nous est satisfaisante. Aussi, il ne paraît pas intéressant d'avoir plus de 8 clusters dans notre cas.

Nous avons essayé nos traitements avec et sans prise en compte des heures jouées. La prise en compte des heures jouées nous semble plus intéressante ; en effet on remarque une grande concentration de joueurs dans un cluster (Figures 1 à 4).

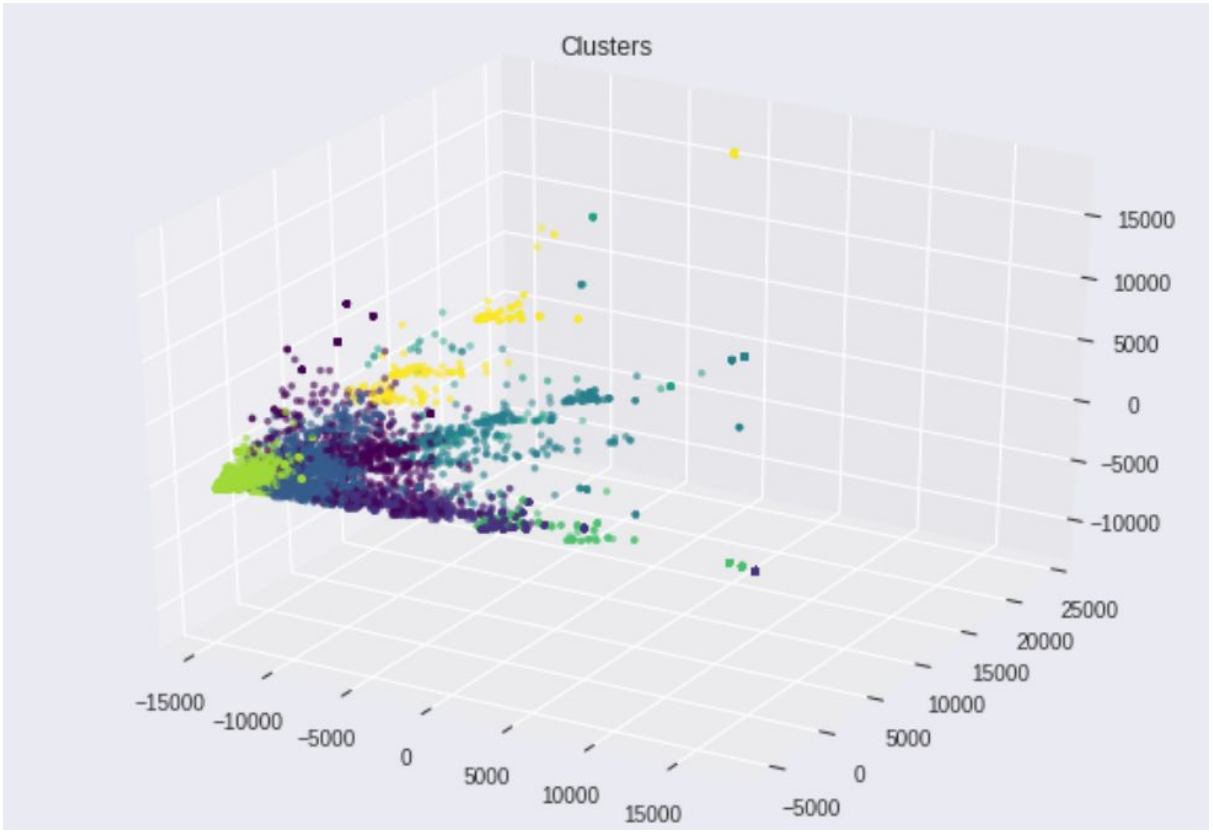

Figure 1 : Clustering sur les Tags avec algorithme Birch (threshold = 0.05, 8 clusters) sans pondération via heures de jeu.

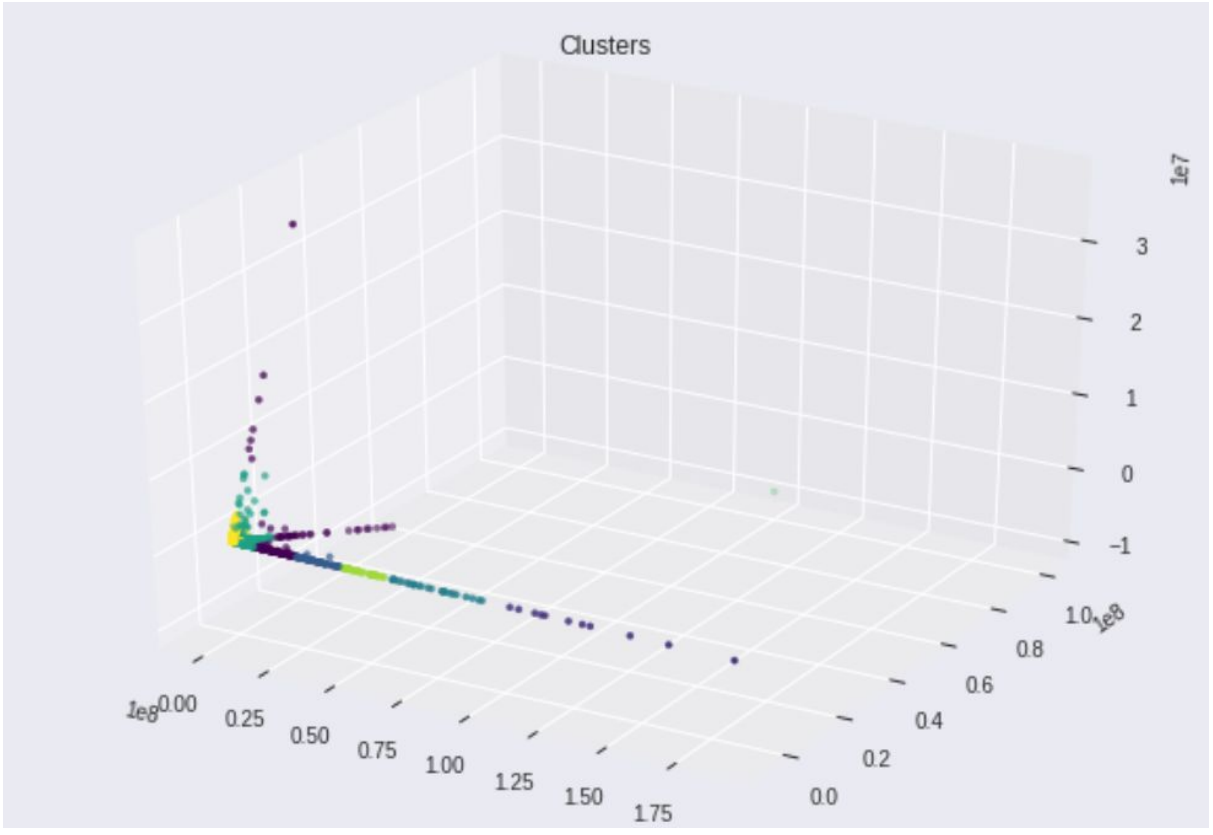

Figure 2 : clustering sur les Tags avec algorithme Birch (threshold = 0.05, 8 clusters) et pondération via heures de jeu.

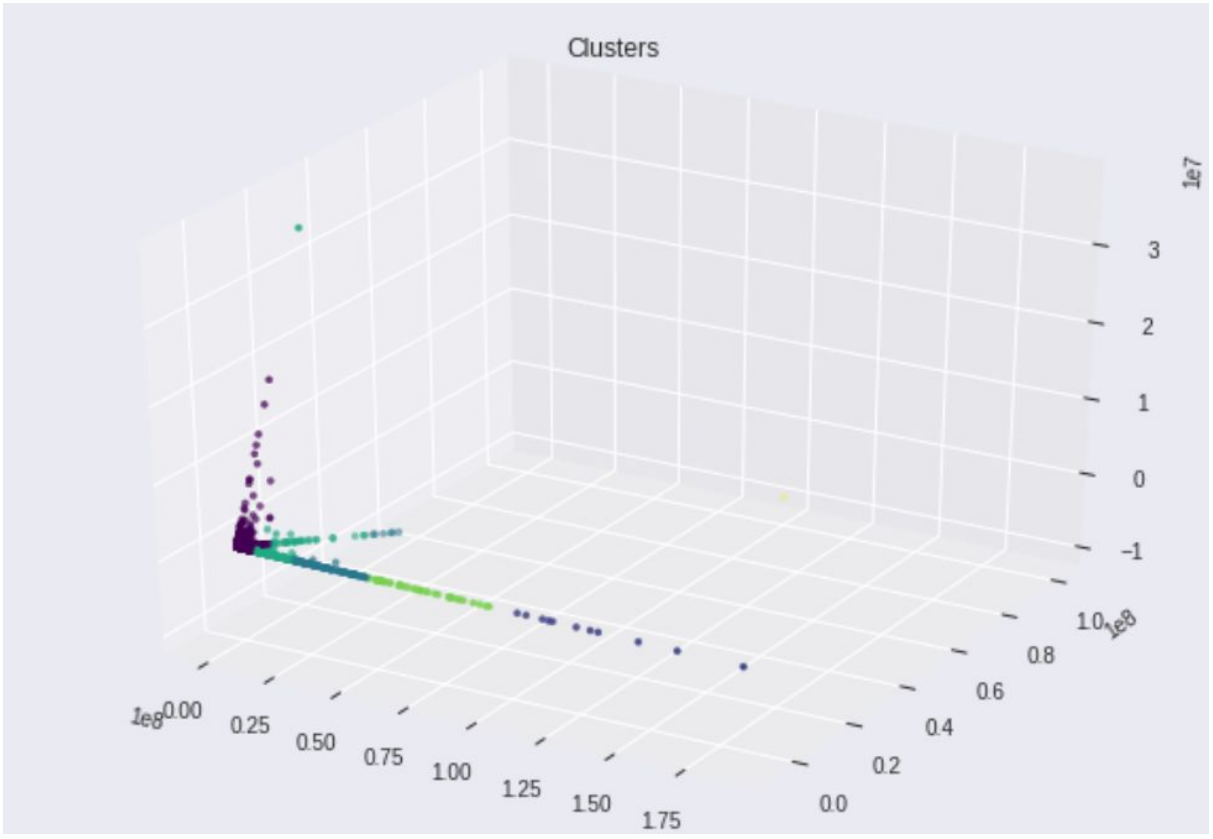

Figure 3 : clustering sur les tags avec algorithme Knime (8 clusters) avec pondération via heures de jeu.

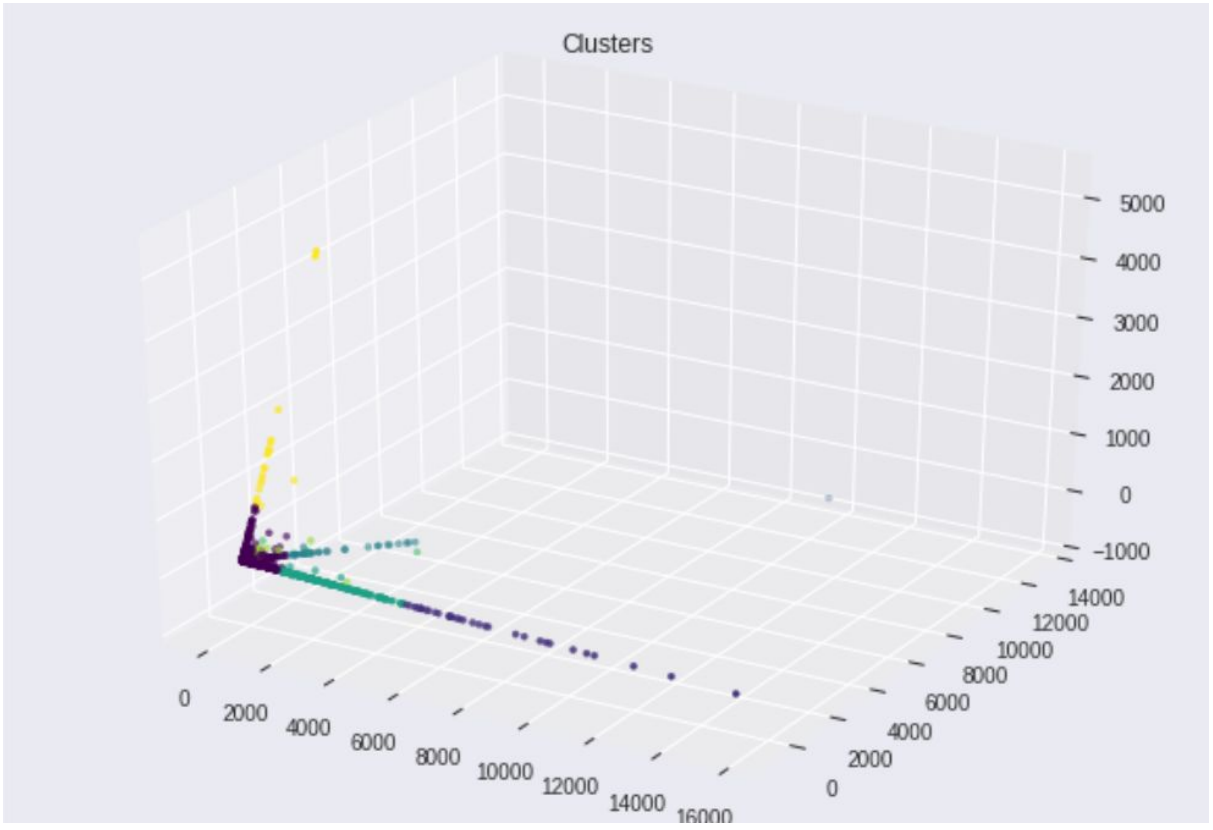

Figure 4 : clustering sur les catégories avec algorithme Knime (8 clusters) et pondération via heures de jeu (résultat très similaire pour l'algorithme Birch).

En détails nous obtenons les clusters suivant (pour les catégories, avec K-means) :
Cluster 0 : 9765 joueurs.
Cluster 1 : 1 joueur.
Cluster 2 : 8 joueurs.
Cluster 3 : 9 joueurs.
Cluster 4 : 33 joueurs.
Cluster 5 : 250 joueurs.
Cluster 6 : 34 joueurs.
Cluster 7 : 21 joueurs.

Les 3 catégories les plus importantes pour chaque clusters sont :

Cluster 0 :
Multi-player, Steam Trading Cards and Valve Anti-Cheat enabled.
Cluster 1 :
Captions available, Includes level editor and Steam Workshop.
Cluster 2 :
Steam Achievements, Single-player and Multi-player.
Cluster 3 :
Single-player, Multi-player and Steam Achievements.
Cluster 4 :
Steam Trading Cards, Multi-player and In-App Purchases.
Cluster 5 :
In-App Purchases, Multi-player and Steam Trading Cards.
Cluster 6 :
Multi-player, Steam Achievements and Steam Trading Cards.
Cluster 7:
Valve Anti-Cheat enabled, Multi-player and Online Multi-Player.

On remarque un cluster très important (et très concentré), ce cluster reste important même si on modifie notre méthode de clusterisation ou si on change agressivement les paramètres. La catégorie “Anti-Cheat enabled” signifie qu’il y a un mécanisme anti-triche pour le jeu. C’est une précondition importante pour les jeux vidéo multijoueurs compétitifs. On peut également voir la catégorie “Multi-player”. D’après nos connaissances nous pouvons confirmer que le jeu multijoueur compétitif est une tendance importante du jeu-vidéo pc actuel.
Le cluster le plus faible est le cluster 1. D’après nos connaissances également, il s’agit d’un profil très rare (“Captions available” signifie que la personne joue souvent à des jeux nécessitant des sous-titres. Cela peut signifier que le joueur est mal-entendant ou qu’il joue à des jeux étrangers ne disposant pas de doublage dans sa langue native. Il joue également à des jeux disposant d’un éditeur de niveaux et également des jeux d’atelier collaboratif.

Proposition d’un ou plusieurs jeux à partir d’un profil

On effectue la moyenne des catégories pour chaque cluster. On trie par ordre décroissant de score pour chaque catégorie et on prend les 3 premiers.
On effectue une succession de traitements (essentiellement des jointures, le détail se trouve dans la dernière partie du Jupyter notebook “gamer profiling on categories”) pour associer les jeux aux clusters les plus pertinents. Ainsi on pourra recommander ces jeux aux joueurs appartenant au cluster (si le joueur ne possède pas déjà le jeu). Pour un joueur donné, il suffit donc de récupérer le cluster auquel il appartient et de connaître les jeux qu’il possède pour lui faire des suggestions. Voici un exemple (Figure 5) du déroulement du programme de recommandation (on limite arbitrairement les recommandations à 3 jeux).

Enter gamer id:
>5250
How many games you want? (maximum number)
>3

You already own:

Team Fortress 2
Dota 2

We recommand:

Left 4 Dead 2
Counter-Strike: Global Offensive
Killing Floor

Figure 5 : exemple de trace d'exécution du programme de recommandation de jeu.

Conseil de jeux à développer pour les développeur

Algorithme

La première piste que nous avons explorée pour définir des groupes de jeux ayant bien marché a été de clusteriser nos données sur KNIME. Cependant les résultats ne nous ont rien apporté. En voici quelques visualisations (Figures 6, 7 et 8).

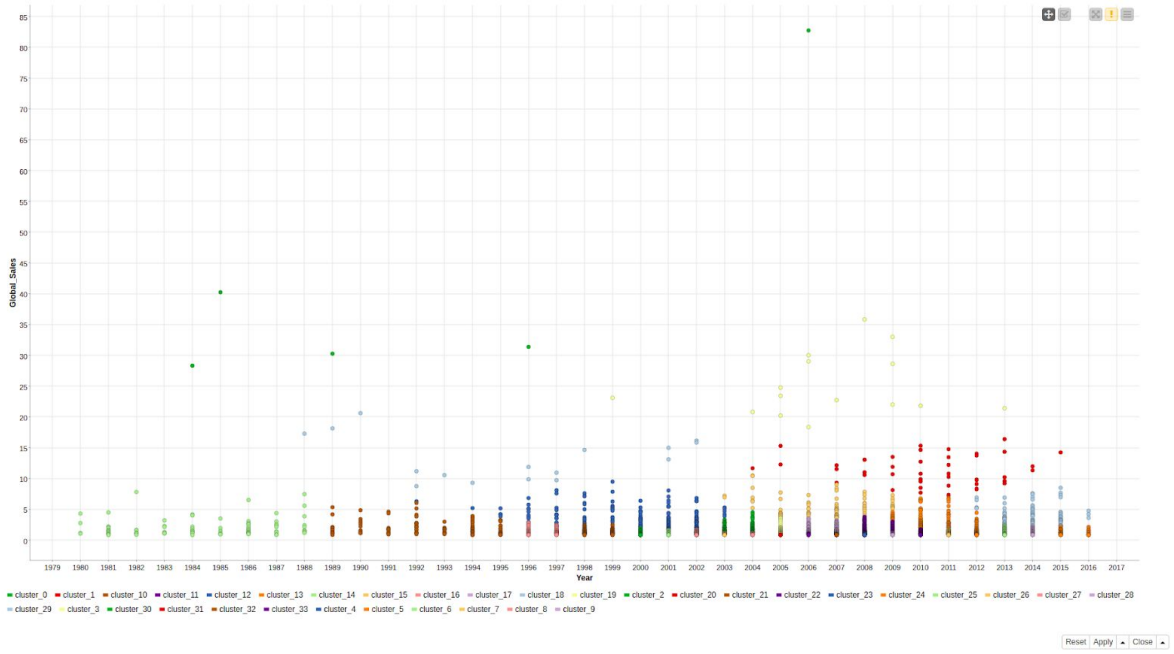

Figure 6 : *clusterisation année/ventes globales avec k-means (34 clusters, une couleur = un cluster).*

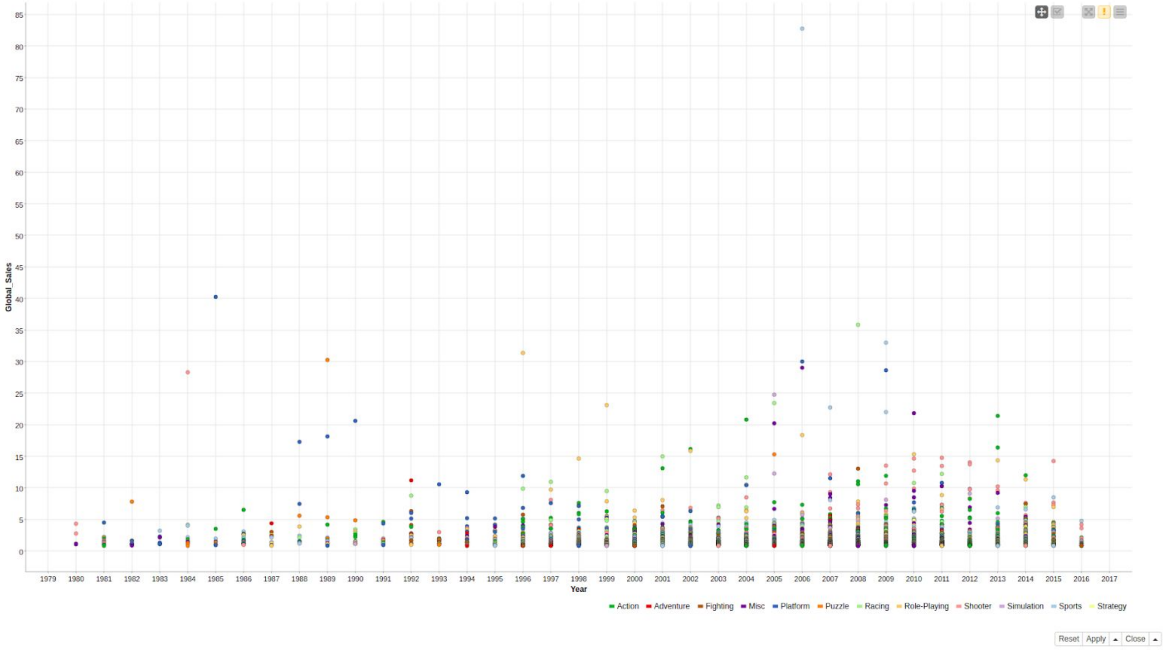

Figure 7 : graph Année/Ventes globales mais les couleurs correspondent aux genres.

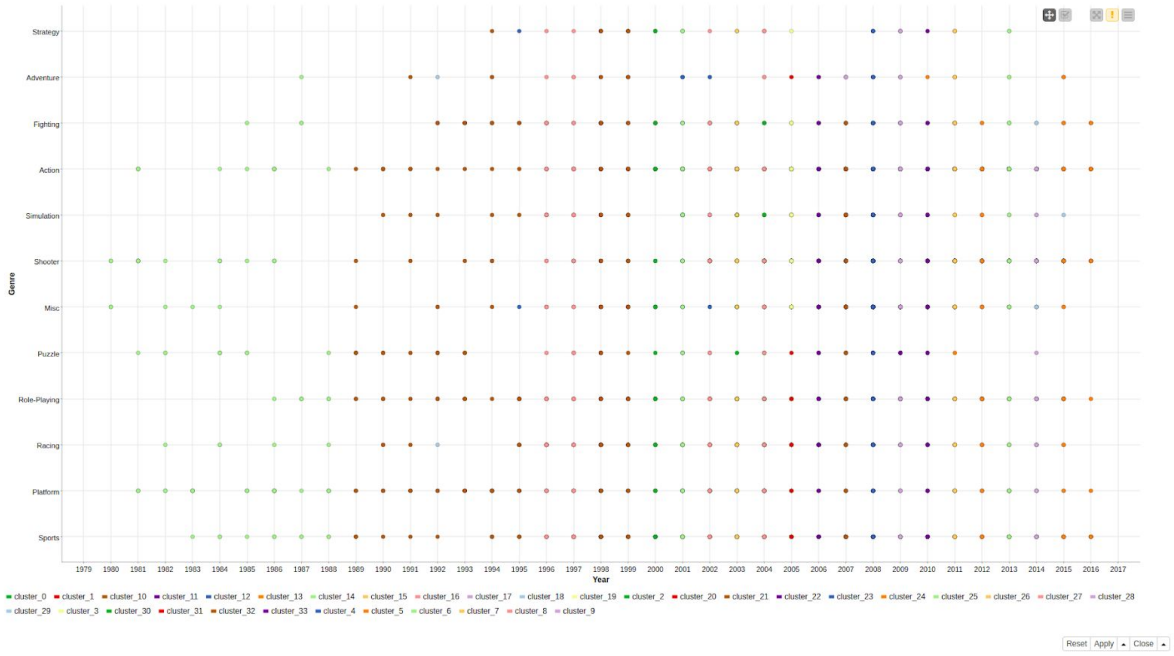

Figure 8: graph Année/Genre et la coloration correspond à un cluster (k-means, 34 clusters).

Comme on peut le voir ces données ne nous ont pas apporté d'informations supplémentaires (pas assez de données pour que le clustering devienne intéressant), nous avons donc décidé d'approcher ce problème avec une analyse de motifs fréquents.

Pour cela nous avons donc généré un nouveau dataset recoupant les données pertinentes provenant des deux jeux de données que nous exploitons (à savoir *vgsales.csv* provenant de https://www.kaggle.com/gregorut/videogamesales et *steam.csv* provenant de https://www.kaggle.com/nikdavis/steam-store-games), cela nous a permis d'avoir un dataset à la fois complet et concis.

Les champs qui nous intéressent sont :

- La plateforme sur laquelle est sorti le jeu (*platform*).
- Le genre du jeu (genre) comme les *Shooter* ou les jeux de rôle par exemple.
- L'année de la sortie du jeu.

Ce dataset sera ensuite parcouru pour en extraire les jeux dont l'année de sortie est postérieure à l'année 2000. En effet nos connaissances sur le sujet nous poussent à penser que c'est après cette date que les consoles de salon sont devenues suffisamment populaires pour effectuer une analyse de marché pertinente.

Notre algorithme est le suivant :

- Extraire toutes les données pertinentes du dataset.
- Lister les différents motifs possibles.
- Supprimer les motifs qui ne vérifient pas la condition de min-support.
- Définir les ventes mondiales moyennes pour chaque motif.
- Trier la liste des motifs.
- Afficher les motifs.

À ces étapes nous avons ajouté plusieurs modifications que vous retrouverez dans notre fichier notebook, à savoir :

- Sélectionner l'année à passer en revue
- Plusieurs modes de recherche en fonction de la taille des motifs valides

Utilisation

Au lancement de notre programme vous aurez donc le choix d'afficher ou non l'état du dataset et des fréquents qui sont extraits à chaque étape de l'algorithme.

Ensuite vous pourrez choisir entre trois modes d'exploration :

1. Genre.
2. Genre + Plateforme.
3. Genre + Plateforme + Année.

De toutes les combinaisons possibles de nos champs nous n'avons retenu que celles-ci qui semblent être les seules combinaisons pertinentes pour définir des tendances sur chaque année ou générales.

Par exemple, voici une exécution de notre programme (Figure 8).

```
Voulez-vous afficher les détails des étapes de l'algorithme (o/n) ?
n
Quelle combinaison de paramètres attendez vous :
1 - Genre
2 - Genre + Plateforme
3 - Genre + Plateforme + Année
3
Quelle année (2000-2016) voulez vous étudier ?
2014
Combien voulez vous afficher de combinaisons parmis les plus vendues ?
3
Les combinaisons les plus vendues sont :

Combinaison : ('PS4', 'Action', '2014')
 ventes moyennes : 1.38 millions
 support du motif : 31
Combinaison : ('XOne', 'Action', '2014')
 ventes moyennes : 0.79 millions
 support du motif : 21
Combinaison : ('PS3', 'Action', '2014')
 ventes moyennes : 0.44 millions
 support du motif : 29
```

Figure 9 : trace d'exécution du programme d'exploration.

Dans cet exemple d'exécution nous pouvons voir que pour l'année 2014 les 3 meilleures combinaisons ont été :

- PS4 : Action.
- Xbox One : Action.
- PS3 : Action.

Ces résultats nous permettent d'avoir une idée des tendances du moment, tendances qui auront inscrit un certain effet de mode puisque pour l'année 2015 les jeux d'actions sont encore majoritairement présents dans le top 5 (4 éléments sur 5) mais ne figurent plus à la première place du "podium".

Pour aller plus loin

Nous aurions aimé avoir des datasets plus récents (idéalement mis à jour en temps réel) mais cela aurait demandé un investissement financier et humain plus important que possible dans le cadre de ce projet. Ces datasets en temps réel auraient permis de faire des analyses de l'année courante afin de pouvoir être au plus près de la tendance actuelle.

Nous avions aussi insisté sur la partie clustering avec un algorithme Python (donc en dehors de la première tentative KNIME) où nous avons essayé plusieurs paramètres de clustering K-means.

Conclusion

Ethique

Un tel projet pourrait porter un grand intérêt commercial. L'utilisateur a conscience qu'il utilise une intelligence artificielle qui va lui conseiller des jeux. Même s'il peut se douter que les choix de l'intelligence artificielle peuvent être biaisés ou même volontairement influencés par un investisseur privé qui aurait financé ce projet (comme un producteur de jeux vidéos), il est important de se demander si ce genre de publicité ciblée ne se ramène pas à un abus de la confiance ou de la crédulité des utilisateurs, avant de rendre une telle plateforme publique. D'autre part, les données collectées proviennent toutes de bases de données rendues publiques, et dont la collecte a fait part d'un consentement des utilisateurs.

Technique

Nous avions donc pour but de générer des conseils pour l'achat ou le développement d'un jeu vidéo, ces objectifs ont dans la mesure du possible été réalisés. Il aurait été idéal de pouvoir approfondir nos méthodes algorithmiques et comme souvent dans l'analyse de jeux de données, avoir des données plus diversifiées et fournies.

Nous avons détaillé chaque étape de nos différents algorithmes dans des fichiers Python notebook. Le code responsable de l'exécution se trouve lui dans un fichier Python à part.