# RAPPORT IAS

## INTRODCUTION:

Le sujet que j'ai choisi, soit une analyse de match Dota 2,une jeu qui se joué en équipe de 5 joueurs, chaque joueur contrôle un personnage appelé "héros" qui possède des compétences et des caractéristiques uniques. Il y a 2 camps "Dire" et "Radiant" qui s'affrontent. L'objectif est de détruire la base ennemie située à l'extrémité opposée de la carte. Je suis très familier avec ce jeu.

Le dataset contient plus de 50 000 matchs joués entre novembre 2015 et mai 2016. Chaque match est décrit par plus de 100 attributs, tels que les héros choisis par chaque équipe, les statistiques de jeu pour chaque joueur, les temps de jeu pour chaque compétence, les événements du jeu tels que les éliminations et les morts de héros, les messages du chat et encore pleins de données.

Mon objectif de base était de prédire simplement qui va gagner une partie par rapport à une équipe de 5. Au final j'ai fait ça et 3 autres travaux:
-Héros les plus/moins joués
-Héros avec le meilleur/pire winrate
-Voir quels sont les mots les plus utilisés dans le chat textuel

## 1ER OBJECTIF: Héros

Après avoir regardé les données, j'ai remarqué un problème dans le csv hero_names, il manque l'id 24, donc pour régler cela j'ai juste fait une petite fonction qui décrémente de 1 tous les id supérieurs à 23.

Afin de voir les winrate ou le nombre de partis d'un héros on a besoin de peu de données, on a juste besoin de voir de quel équipe il était et si l'équipe radiant a gagné, ainsi que les infos sur son choix d'héros. Pour simplifier mon code j'ai cré 2 colonnes is_win qui renvoie un booléen et une colonne is_played toutes à 1 pour faciliter le comptage de parties.

Après c'est juste du traitement basique de dataframe pour calculer winrate, c'est à dire diviser le nombre de is_win true pour chaque héros divisé par le nombre de fois qu'il a été joué. Pour récupérer le plus élevé, ou le plus faible, on joue juste avec le paramètre ascending de sort_values.

Pour voir les persos les plus/moins joué il suffit de suivre ce principe mais juste en regardant la colonne is_played.
Ensuite j'ai fait en sorte de les afficher avec des barplots de searborn

Pour voir les résultats de tout ceci, ouvrir le fichier *Results heroes.pdf*

# 2EME OBJECTIF: Chat

Tout d'abord, j'ai vérifié le nombre de messages différents et je vois assez vite que c'est une catastrophe, 91% des messages apparaissent qu'une seule fois. Du coup, je me dis tout simplement que des mots ont été écrit avec d'autres mots par exemple "gg" accompagné d'un "dude" ou "mate". Donc pour régler cela je fais une question qui regarde autour de certains mots fréquemment utilisés. Après avoir fait tourné sur tous les messages cette fonction ( elle prend pas mal de temps), je remarque que la répartition des messages est incroyablement mieux géré, seulement 4,4% de messages apparus qu'une fois .

Afin d'avoir une visualisation de ses données, je me renseigne sur comment l'afficher au début je pense à faire un barplot mais je découvre Wordcloud et dont voici le résultat ( voir *word_result.pdf*)

# 3EME OBJECTIF: Prédiction

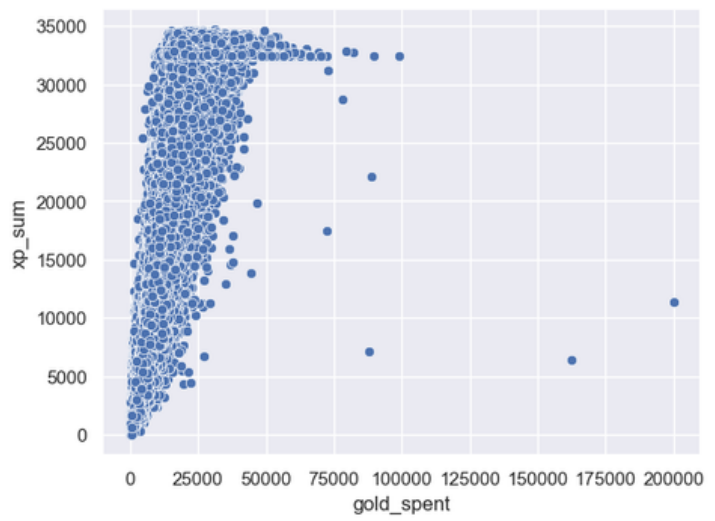
<Au début de mon projet, je n'avais pas remarqué les fichiers test_player et test_labels donc j'ai commencé par m'intéresser par les autres fichiers ainsi pour comprendre leur utilité et ce qui est utile j'ai fait une EDA qui m'a vite permit de remarqué que toutes les données commençant par unit_ étaient inutiles.
J'ai fait en sorte de régler un peu tous les problèmes que  j'ai trouvé dans le dataset comme le fait que les dates dans le fichier patch_dates

étaient au format ISO alors que dans le fichier match les dates sont en entier. Une autre erreur est dans le fichier hero_names où il manque le hero 24, en réalité ce n'est pas vraiment une erreur, les parties enregistrés se déroule pour 95% des cas dans un patch mais lors du patch suivant un héros a été "rework".

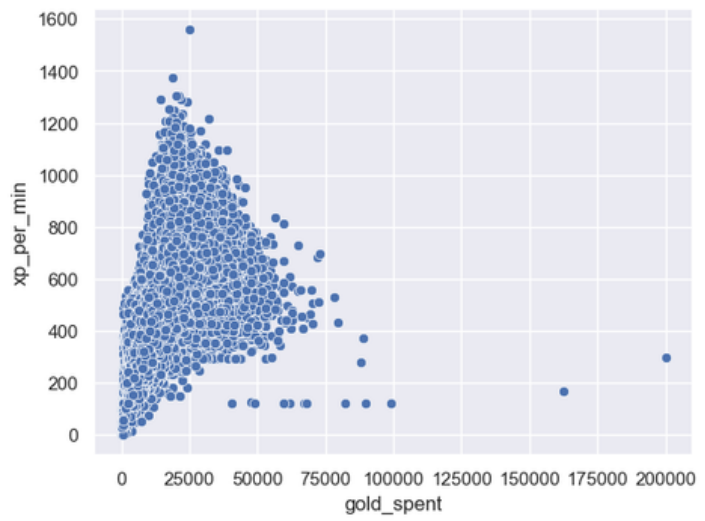
Le code n'est pas vraiment dans l'ordre mais j'ai préféré le garder tel quel car cela montre ma démarche et par où je suis passé avant de faire de la prédiction.

Du coup, à un moment je me dis est-ce que se serait bien de faire une PCA pour voir des mauvais cas à éliminer mais je me suis dis autant essayer d'en faire une "manuelle" donc je démarre avec 4 scatterplots de seaborn et j'observe assez rapidement quelques points éloignés et qui ne sont pas du tout raccord avec le graphe. C'est à ce moment que j'aimerais bien avoir votre avis sur ce moment aurais-je du les retirer ou pas:

x: gold_spent
y: xp_sum




x: gold_spent
y: xp_per_min




# Preprocessing:

Pour démarrer le preprocessing on crée un dataframe contenant la compo des matchs issu des données de test_player et hero_names. Pour chaque match, à partir du match_id on regarde dans le fichier test_labels si radiant a gagné ce match ou pas. Comme toujours on vérifie si on a des données inutiles, ce qui n'est pas le cas ici.

Pour permettre d'utiliser des modèles je crée 4 tableaux contenant les équipes sous forme de id et de nom. On crée ainsi une liste contenant les 2 équipes.

Maintenant que nos données sont utilisables on peut commencer à utiliser des modèles:

# Méthode(1/3): Random Forest

Le Random Forest est un modèle très puissant lorsqu'on est dans une situation avec des choix, par exemple il aurait été l'un des meilleurs si notre objectif était de savoir quel était les meilleurs items à acheter ou bien quelle capacité à augmenter. Cependant il est toujours assez intéressant dans notre cas.

| prediction | 0 | 1 |
|---|---|---|
| actual | | |
| 0 | 10089 | 9161 |
| 1 | 7799 | 12945 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.56 | 0.52 | 0.54 | 19250 |
| 1 | 0.59 | 0.62 | 0.60 | 20744 |
| accuracy | | | 0.58 | 39994 |
| macro avg | 0.57 | 0.57 | 0.57 | 39994 |
| weighted avg | 0.58 | 0.58 | 0.57 | 39994 |

# Méthode(2/3): KNeighboors

K-Nearest Neighbors est un modèle très puissant dans le cas où les exemplaires d'entrainement sont similaires, dans notre cas on doit avoir des combinaisons qui arrivent mais dans le cas d'une partie où les niveaux sont totalement distincts. Le résultat sera biaisé, si on avait pu utilisé le trueskill, les prédictions auraient surement été meilleurs.

| prediction | 0 | 1 |
|---|---|---|
| actual | | |
| 0 | 9151 | 10099 |
| 1 | 7434 | 13310 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.55 | 0.48 | 0.51 | 19250 |
| 1 | 0.57 | 0.64 | 0.60 | 20744 |
| accuracy | | | 0.56 | 39994 |
| macro avg | 0.56 | 0.56 | 0.56 | 39994 |
| weighted avg | 0.56 | 0.56 | 0.56 | 39994 |

# Méthode(3/3): Régression logistik

Une des méthodes les plus utilisés pour un cas tel que le notre, soit prédire une variable binaire. Ce modèle est là pour créer des liens entre les variables.

| prediction | 0 | 1 |
|---|---|---|
| actual | | |
| 0 | 10377 | 8873 |
| 1 | 7352 | 13392 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.59 | 0.54 | 0.56 | 19250 |
| 1 | 0.60 | 0.65 | 0.62 | 20744 |
| accuracy | | | 0.59 | 39994 |
| macro avg | 0.59 | 0.59 | 0.59 | 39994 |
| weighted avg | 0.59 | 0.59 | 0.59 | 39994 |

# AMELIORATIONS:

Une des premières qui me vient en tête serait de prendre en compte qui fait le premier kill, qui est un évènement assez important dans une partie de Dota 2.

Un autre moyen aurait été si on pouvait exploiter trueskill ou alors avoir le rang de chaque joueur dans la partie.

J'aurais aussi pu faire un projet sur les choix d'items pour chaque personnage en ayant comme sorti d'affichage un decision tree issu d'un random forest pour avoir les meilleurs résultats possibles avec des pourcentages et une suite intéressante par rapport à cet item