Année universitaire 2021/2022

Master 1 DAC

PROJET :

League of Legends : Analyse des parties des meilleurs joueurs du monde entier

Business Intelligence & User Modelling

Enseignant : Laure Soulier

Valentin FONTANGER - 21115619

Enzo Durand - 21107517

Dan Serraf – 3971120

Table des matières

Introduction

En 2021, les matchs professionnels du jeu League Of Legends ont été visionné plus de 664.16 millions d'heures par des auditeurs du monde entier (en excluant la Chine et l'Inde). Le jeu a également établi un nouveau record d'audience lors de la finale de la compétition World Championship avec un total de 4 millions de fans. D'après un rapport publié par SuperData, la société Riot Games auraient généré 1.75 milliards de dollars en 2020, contre 1.5 en 2019. League of Legends est l'un des jeux les plus visionnés au monde. Bien que complexe, c'est avec passion que les fans tentent de recopier le comportement des pros dans le but de remporter le plus de parties.

Afin de mieux comprendre le comportement en jeu des meilleurs joueurs, nous avons réalisé une analyse en profondeur de leurs habitudes, leurs scores ainsi que leurs tactiques, en portant une attention particulière à leur nationalité et serveur de jeu dans le but de d'observer des différences et similitudes. Ce travail s'inscrit dans une double problématique.

Dans un premier temps, les actions et décisions des meilleurs joueurs du jeu reflètent, par le biais du streaming (méthode de diffusion de contenu en temps réel), sur les joueurs lambdas. Il est donc important de prévoir les tendances ou éventuels changements qui s'opéreront au cours de la saison, par pays et par serveur. Par exemple, si un joueur professionnel met en place une nouvelle tactique, on peut s'attendre à un changement de **méta** (le méta est le style de jeu adopté par la communauté). On peut dès lors prévoir les rééquilibrages et modifications à apporter au jeu. Dans un second temps, il est important de déterminer les différents niveaux des joueurs en parties classées. Il existe une disparité entre les joueurs, et cette dernière demande une analyse approfondit afin d'adapter ou de créer de nouvelles divisions et sousdivisions, permettant une homogénéité des niveaux de jeux.

Nous avons prêté une attention particulière aux différents niveaux de granularités géographiques durant notre travail. Nous avons souligné les différences de comportements et d'actions entre les différents pays.

Tout d'abord, nous détaillerons le processus d'acquisition des données. Dans un second temps, nous exposerons les différentes sous problématiques et les schémas de données permettant de les résoudre, ainsi que les résultats observés. Finalement, nous expliquerons les méthodes de Machine Learning employés pour analyser les différentes variables dans un contexte de prédiction de victoire.

1. Acquisition de données

1.1 Mécanisme d'extraction des données

Notre travail d'acquisition de données s'articule en plusieurs étapes : une étape de scraping et une étape de stockage. Nous avons utilisé deux sources de données, **l'API riot games** : (https://developer.riotgames.com/) et **trackingthepros.com.** L'API riot est très complète et bien documentée. Elle fournit une importante quantité d'information sur les joueurs, les différentes divisions et les détails des parties jouées. Le site trackingthepros est géré et administré par des joueurs passionnés, vérifiant les informations sur les joueurs, tout en les mettant à jour régulièrement. Ce site est une référence dans le domaine compétitif de League of Legends.

Dans un premier temps, nous avons effectué un scrapping du site trackingthepro.com afin de **récupérer les meilleurs joueurs du jeu et leurs nationalités**, information que l'api ne fournit pas, l'api ne fournissant uniquement les zones géographique (Europe de l'ouest, du nord, brésil, Corée, Amérique du nord, …). Une fois les joueurs et nationalités récupérées, nous obtenons les pseudonymes des différents comptes qu'ils possèdent. Une fois ces pseudos en notre possession, nous utilisons l'api afin de récupérer l'identifiant, le **puuid**, associé au pseudo, ainsi que les dernières parties jouées. Nous avons volontairement exclu les parties issues de la saison 11, saison précédente du jeu qui s'est achevée en janvier 2022, afin de prendre en compte les différentes mises à jour du jeu, mais également l'état d'esprit des joueurs, plus compétitifs en début de nouvelle saison.

Pour cette étape, nous avons utilisé la librairie python **Selenium** afin d'automatiser le navigateur web, **BeautifulSoup** afin de récolter les informations des pages webs et **requests** pour effectuer les requêtes.

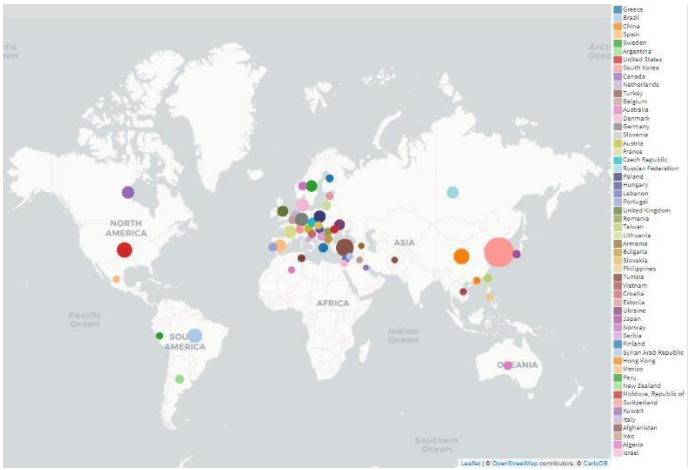
1.2 Preprocessing et Nettoyage

Nous avons procédé à un nettoyage et preprocessing des données récoltées. Nous disposons de deux jeux de données, un listant les joueurs, et l'autre, leurs matchs.

Concernant la premier, nous avons récolté l'ensemble des comptes liés à un joueur. Il y a donc autant de lignes dans notre table, que de comptes. De nombreux joueurs ne disposaient pas d'une information exacte sur la localisation de leur serveur de jeu, nous avons donc fait le choix de les supprimer de la liste de joueurs.

La colonne âge des joueurs comportait des valeurs manquantes, que nous avons remplacé par l'âge moyen des joueurs.

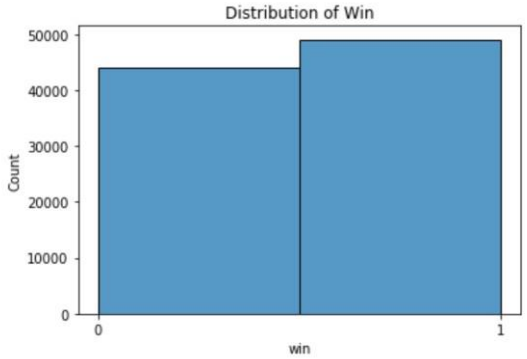
Concernant le second jeu de données, nous avons éliminé les colonnes vides (riotIdName, riotIdTagline). Dès lors, nous possédons 117 colonnes. Nous avons remplacé certains noms de pays tels que « Russian Federation » en « Russian », afin que Dataiku puisse traiter cette donnée géographique.Finalement, on fusionne les deux tables à l'aide d'une jointure de type « inner ». Nous obtenons finalement un jeu de données avec des joueurs dont les comptes se trouvent dans les deux tables. En plus des données du jeu, nous procédons au traitement des données issues de dataset sociologiques. Nous avons supprimé les colonnes inutiles et avons converti toutes les dates en années. Nous procédons ensuite à un group by au niveau des pays et années. Nous complétons les valeurs manquantes par des valeurs cohérentes pour chaque colonne.



Répartition des joueurs du jeu de données à travers le monde

Nous obtenons un dataset relativement équilibré :

```
Win rate over the dataset : 52.71 %
Loose rate over the dataset : 47.29 %
```



Histogramme de la variable victoire

2. Axes d'études et méthodologies

2.1 Thématiques sous-jacentes

La question centrale de ce travail est la suivante : **Comment et pourquoi gagne-t-on un match de League Of Legends à très haut niveau** ? Nous rappelons l'importance de répondre à cette question cruciale :

- Prévoir les changements de **tactiques** et de **comportement** des joueurs en général et par **pays**.
- Prévoir les potentiels **rééquilibrages** du jeu
- Créer un meilleur système de **matchmaking** (système permettant de constituer des équipes de niveaux homogènes).

Afin de répondre à cette problématique, nous introduisons 3 fais importants : Victoire, Dommage et Stratégie. Ces faits nous permettent d'obtenir des informations importantes répondant à nos sous problématiques. Voici le schéma en constellation réalisé avec Google Drawing.

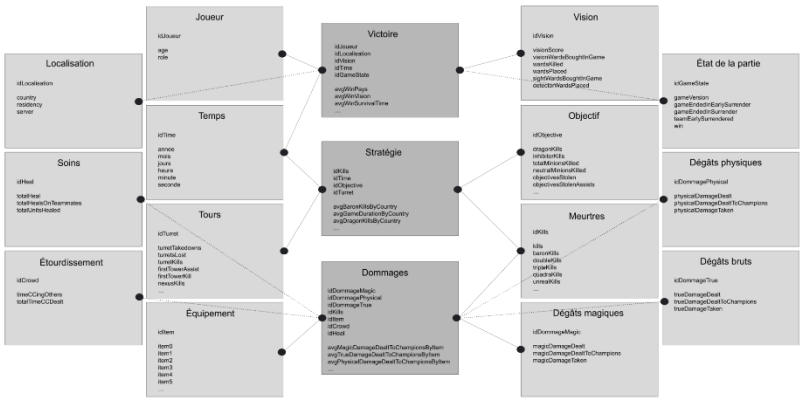


Schéma en constellation

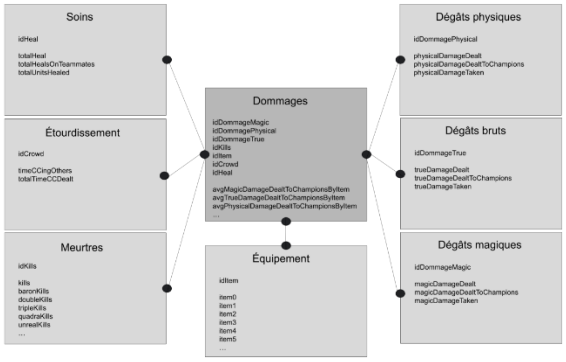


Schéma en étoile du fait Dommages

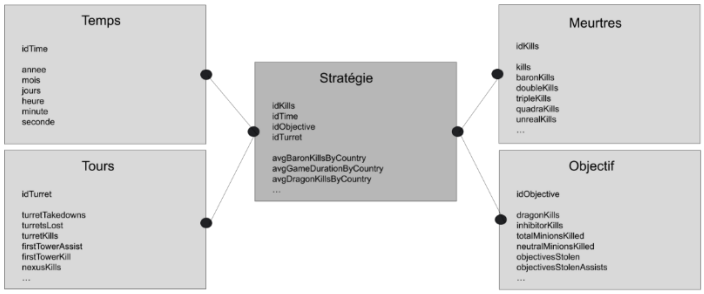


Schéma en étoile du fait Stratégie

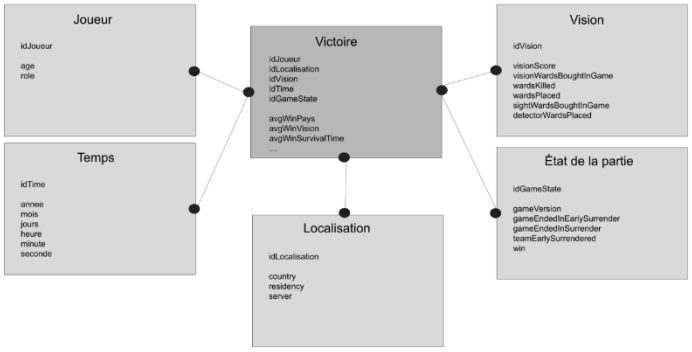


Schéma en étoile du fait Victoire

Le fait Victoire nous permet de consulter les statistiques nationales et continentales sur le comportement au niveau de la vision en jeu, du temps de jeu, et l'âge des joueurs. Ce fait se veut général et ne rentre pas dans les détails techniques du jeu, à l'exception de la dimension vision. Le fait Stratégie nous permet d'accéder aux informations concernant le comportement tactique des joueurs durant les parties par pays et par continent. Finalement, le fait Dommages nous propose de consulter l'impact des objets et des dégâts infligés ou reçus en jeu et leur impact sur la victoire, très utile pour observer les objets tendances ou au contraire délaissés.

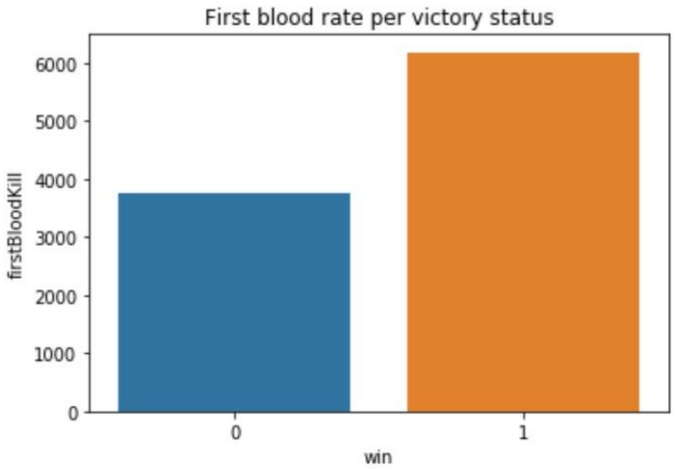
3. Démarche Machine Learning

Nous avons souhaité tirer partie du Machine Learning afin de mieux comprendre les ingrédients permettant à un joueur professionnel de gagner. **Nous n'évoquerons dans cette partie uniquement les découverte majeures ou importantes, et laissons le lecteur consulter l'intégralité du notebook en annexe**. Nous avons utilisé la librairie **scikit-learn**, nous donnant accès aux implémentations d'algorithmes d'apprentissage statistiques, et avons réalisé nos visualisations avec **seaborn**.

3.1 Exploration des données

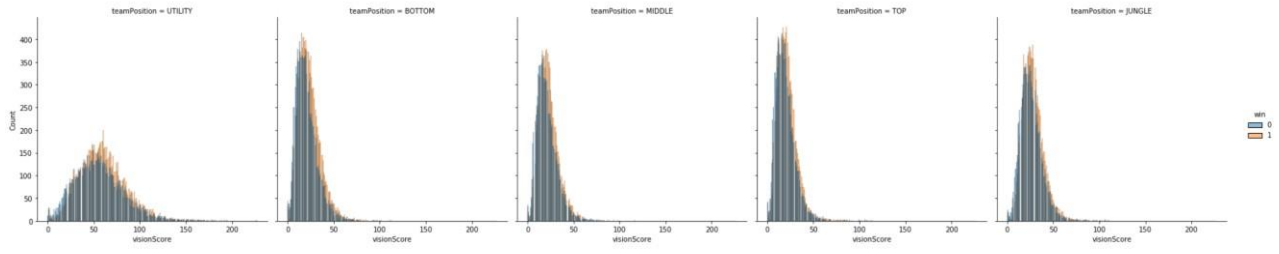
L'exploration de données nous a permis de mieux sélectionner nos variables afin de nourrir notre modèle de Machine Learning. Dans un premier temps, nous procédons à des analyses multivariées entre nos variables et la variable étiquette, dans notre cas, la victoire (colonne « win »).

Nous observons des résultats surprenant concernant l'impact de la première victime. En effet, les joueurs ayant contribués à tuer le premier ennemi ont une probabilité supérieure de gagner.



First blood par statut de victoire

Nous enquêtons également sur l'impact du score de vision sur la victoire du joueur. Nous avons fait l'hypothèse que le score de vision du joueur pour le rôle « support » n'a que très peu d'impact sur la victoire au niveau professionnel, car l'unique support de chaque équipe a pour tâche de constamment regarder la carte et apporter la vision à son équipe. On s'attend en effet à ce que les autres rôles de chaque équipe face la différence en apportant également la vision à leur équipe, bien que cette tâche ne soit pas la plus importante pour eux.



Distribution du score de vision par rôle et par statut de victoire

Pour vérifier cette hypothèse, nous avons conduit le **test de Kolmogorov-Smirnov** afin de vérifier si les distributions du score de vision conditionnée par la victoire et la défaite suivent la même probabilité. Nous obtenons une P-value largement inférieure à 0.05, nous rejetons

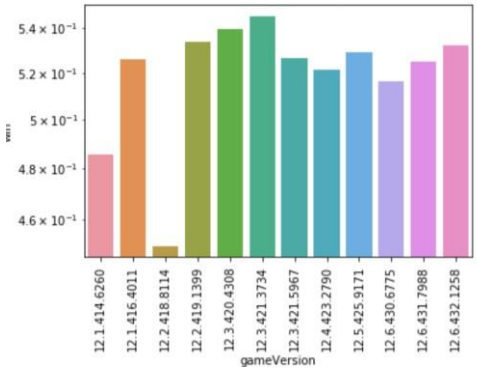
l'hypothèse nulle d'identique distribution en faveur de l'hypothèse alternative. La distribution du score de vision varie selon la victoire ou la défaite.

Nous avons souhaité analyser l'impact de la version du jeu sur les statistiques des joueurs. Voici la répartition des matchs selon les versions :

```
12.5.425.9171    43955
12.4.423.2790    10555
12.1.416.4011     9976
12.3.421.5967     7891
12.6.432.1258     7037
12.2.419.1399     6449
12.6.430.6775     4928
12.1.414.6260      762
12.3.420.4308      547
12.3.421.3734      429
12.6.431.7988      341
12.2.418.8114      229
Name: gameVersion, dtype: int64
```

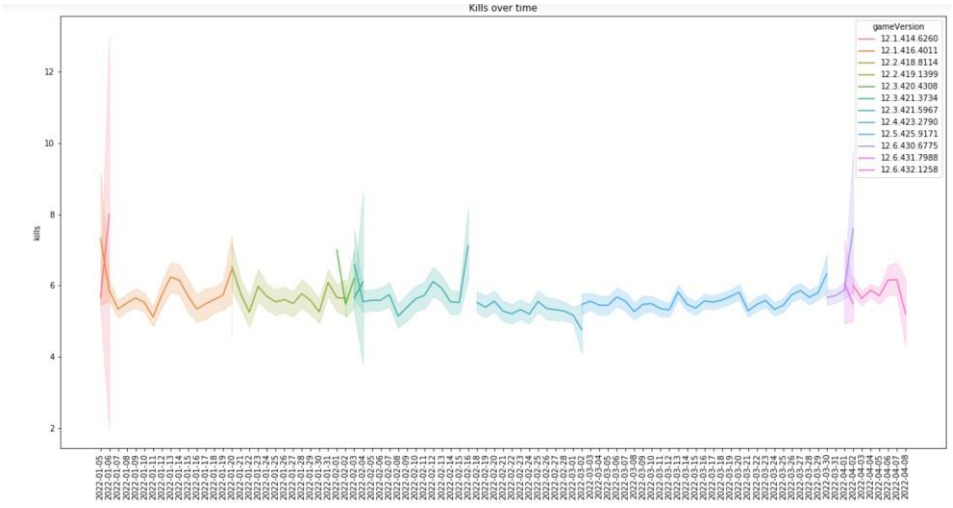
Distribution des matchs par version du jeu

Nous avons, par exemple, analysé le taux de victoire par version



Taux de victoire par version du jeu

Nous procédons également à une analyse temporelle du nombre de kills au courant de la saison 12



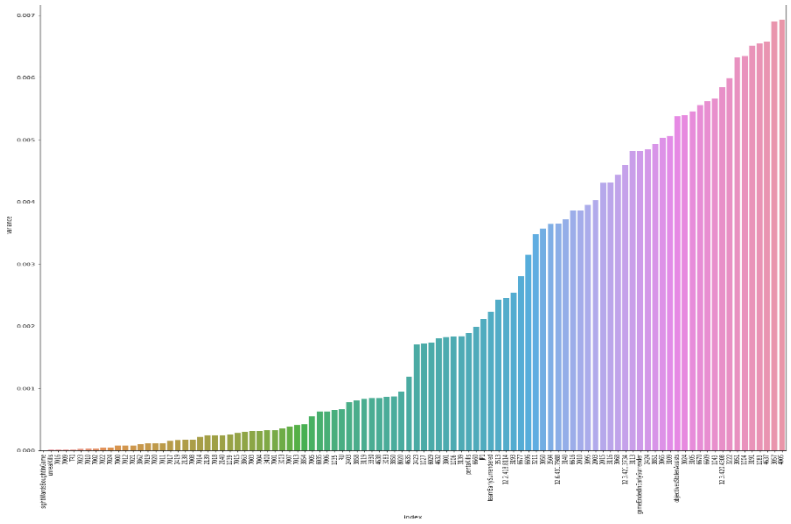
Evolution du nombre de kill moyen au cours du temps

Nous remarquons les nouvelles versions surviennent lorsque le nombre de kills moyen augmente ou diminue de façon plus importante.

Notre dataset comportant de nombreuses variables catégorielles, nous avons procédé à un encodage de type one-hot, augmentant considérablement le nombre de features. Il a donc été primordial de procéder à une sélection des variables afin de combattre le fléau de la dimensionalité.

3.2 Feature Selection

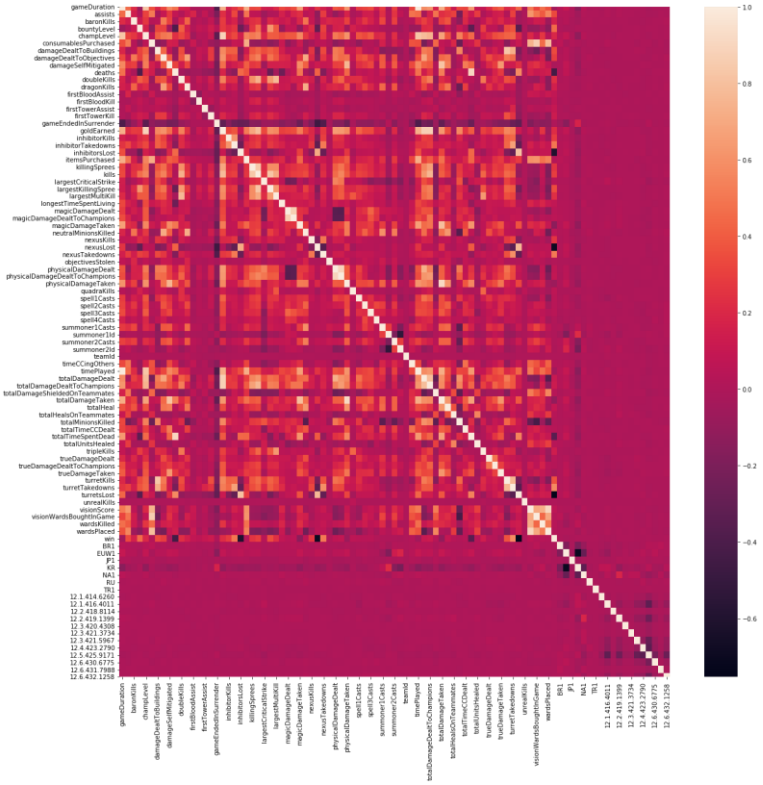
Dans un premier temps, nous avons mis de côté les variables continues avec très peu de variations. Sur l'axe des abscisses, les nombres sont les variables one-hot encodé des équipements du jeu.



Top 100 des variables avec le moins de variance

Dans un second temps, nous avons observé la corrélation entre les variables. Le coefficient de Pearson nous renseigne sur la corrélation et fait l'hypothèse de la continuité des variables. De plus, ce coefficient expose les corrélations linéaires entre variables.

Nous avons ensuite procédé à l'élimination des variables très corrélées en ne gardant qu'une seule d'entre elles à chaque fois. Nous laissons la possibilité au lecteur de lire le processus de sélection plus en détail à l'aide du notebook fournit en annexe.



Heatmap de la corrélation entre les variables

3.3 Entraînement et résultats

Nous choisissons d'entraîner sans aucune optimisation les modèles KNN, Linear SVM, DecisionTree, Random Forest, Naive Bayes et Regression Logistique. Notre ensemble de test est composé de 27930 données tandis que notre ensemble d'entraînement de 65169 données. Chaque point de donnée est décrit par 473 variables.

Nous avons porté une attention particulière à l'importance des variables durant le processus d'entraînement. Voici les résultats obtenus :

```
--- Loading Nearest Neighbors ---
--- Training : done ---
              precision    recall  f1-score   support

           0       0.74      0.67      0.71     14468
           1       0.68      0.74      0.71     13462

    accuracy                           0.71     27930
   macro avg       0.71      0.71      0.71     27930
weighted avg       0.71      0.71      0.71     27930
```

KNN

	precision	recall	f1-score	support
0	0.97	0.96	0.96	13353
1	0.96	0.97	0.97	14577
accuracy			0.97	27930
macro avg	0.97	0.97	0.97	27930
weighted avg	0.97	0.97	0.97	27930

SVM

	precision	recall	f1-score	support
0	0.95	0.94	0.94	13261
1	0.95	0.95	0.95	14669
accuracy			0.95	27930
macro avg	0.95	0.95	0.95	27930
weighted avg	0.95	0.95	0.95	27930

Decision Tree

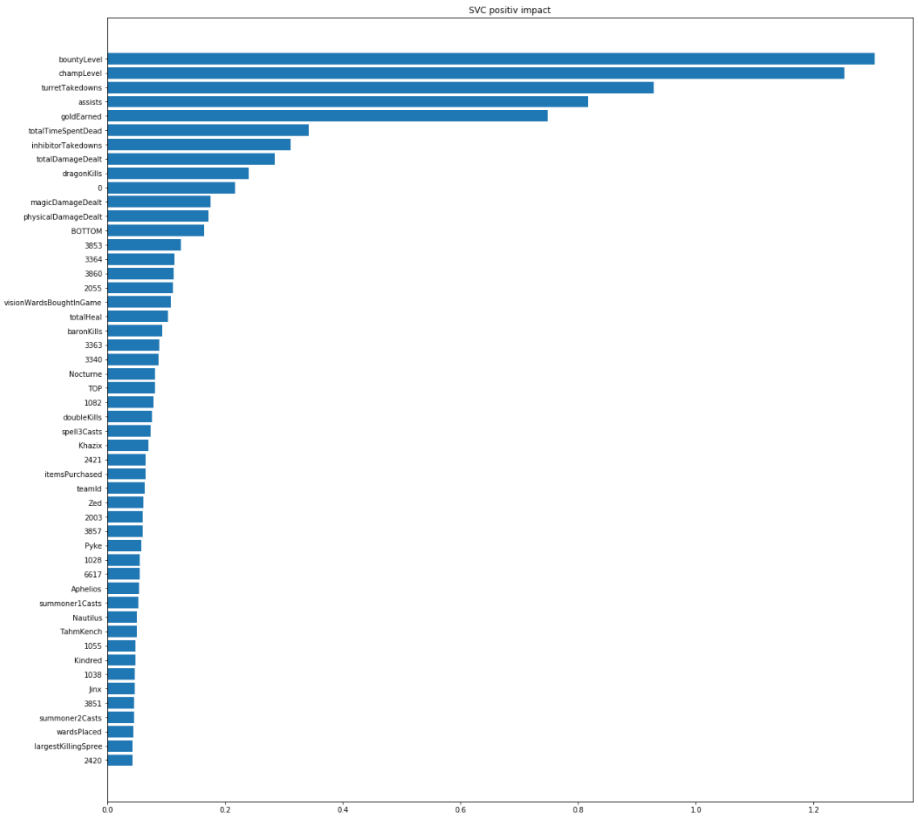
	precision	recall	f1-score	support
0	0.09	0.72	0.16	1613
1	0.97	0.54	0.69	26317
accuracy			0.55	27930
macro avg	0.53	0.63	0.43	27930
weighted avg	0.92	0.55	0.66	27930

Random Forest

	precision	recall	f1-score	support
0	0.94	0.59	0.73	20902
1	0.42	0.88	0.57	7028
accuracy			0.67	27930
macro avg	0.68	0.74	0.65	27930
weighted avg	0.81	0.67	0.69	27930

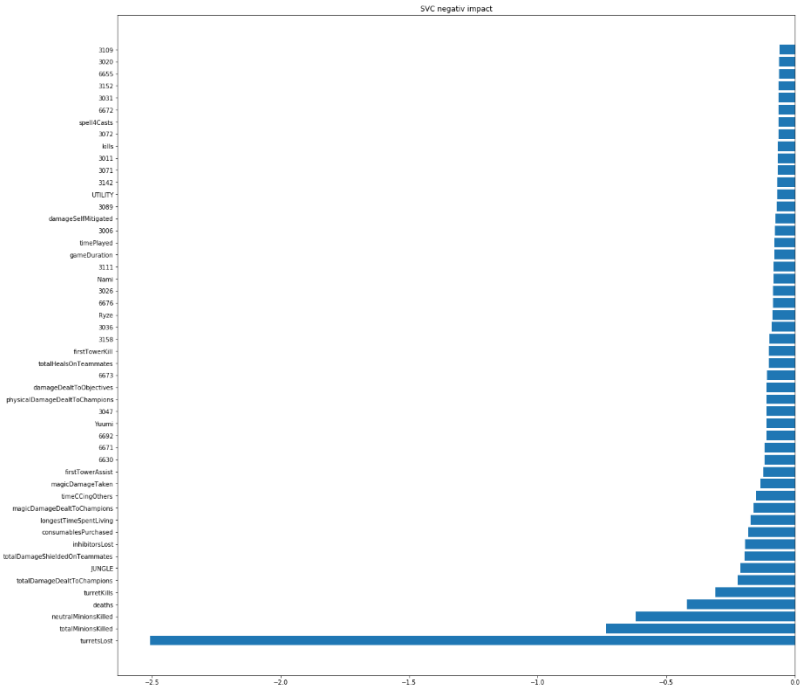
Naive Bayes

Nous avons donc choisi le modèle SVM. Par ailleurs, une étude des poids du modèle nous permet de mieux comprendre les variables impactant positivement et négativement la victoire d'un joueur à un tel niveau :



BountyLevel désigne la prime placée sur le joueur lorsque celui-ci reste en vie pendant une longue période tout en éliminant les joueurs adverses. Ce style de jeu contribue donc positivement à la victoire du joueur. Le nombre de tour détruite et le nombre d'or gagné sont logiquement des facteurs contribuant positivement à la victoire. On retrouve l'objet 3364 (id relié au nom d'objet « Oracle Lens », réputé pour sa puissance et efficacité en jeu. L'objectif dragon est un objectif clef pour s'emparer de la victoire. Les récentes mises à jour du jeu ont contribué positivement à l'accomplissement des objectifs durant la partie. Les objectifs permettent aux équipes soudées d'obtenir des bonus de dégâts, de l'or, mais également de cultiver un état d'esprit positif au sein de l'équipe.

Observons les variables impactant la victoire de façon négative :



On retrouve évidemment le nombre de tours détruites par l'adversaire. Cependant, bien qu'une équipe ait toutes ses toures détruites, cela ne signifie pas qu'elle ne peut plus gagner.

Les résultats obtenus sont encourageants et mettent en avant les combinaisons de variables menant à la victoire.

Dans le cadre de futurs travaux, nous souhaitons procéder à une sélection des variables bien plus poussées. De plus, nous souhaitons acquérir un autre type de donnée : les données chronologiques de chaque partie. En effet, la prochaine étape consiste à prédire la victoire ou non du joueur dès ses 10 à 15 premières minutes de jeu.

Jeu de données des joueurs :

- **Id** : id dans la liste
- **Name** : nom du joueur
- **Role** : le rôle principal en jeu
- **Country** : La nationalité du joueur
- **Residency** : Le pays de résidence du pays
- **Server** : le serveur sur lequel le joueur joue
- **Summoner_names** : Une liste des noms de comptes du joueur. Un joueur peut disposer de plusieurs comptes

Jeu de données des parties :

- **gameCreation :** Timestamp de creation de la partie (en ms)
- **gameDuration :** Durée de la partie (en ms)
- **gameEndTimestamp :** Timestamp de fin de la partie (en ms)
- **gameId :** Identifiant de la partie
- **gameMode :** Mode de jeu
- **gameName :** Nom unique de la partie
- **gameStartTimestamp :** Timestamp de début de la partie (en ms) - **gameType :**
- **gameVersion :** Version du jeu à l'instant de la partie
- **mapId :** Identifiant unique de la partie
- **platformId :** Nom du serveur de jeu
- **queueId :** Identifiant de la division
- **tournamentCode :** Identifiant du tournoi
- **assists :** Nombre de fois ou le joueur à aider à tuer un membre de l'équipe adverse dans la partie
- **baronKills :** Nombre de fois où le joueur à tuer le l'objectif appelé « Baron »
- **bountyLevel :** Nombre de fois où une prime est placée sur le joueur
- **champExperience :** Nombre de points d'expériences acquis sur le champion
- **champLevel :** Niveau du champion à la fin de la partie
- **championId :** Identifiant unique du champion
- **championName :** Nom du champion joué
- **championTransform :** Transformation du champion si possible (seulement sur Kayn)
- **consumablesPurchased :** Nombre d'objets consommables de la boutique achetés en jeu
- **damageDealtToBuildings :** Dégâts infligés aux objects ennemis
- **damageDealtToObjectives :** Dégâts infligés aux objectifs
- **damageDealtToTurrets :** Dégâts infligés aux tours
- **damageSelfMitigated :** Dégâts infligés à soit même
- **deaths :** Nombre de fois où le joueur est mort

-

detectorWardsPlaced : Nombre de fois où le joueur à placé un détecteur sur la carte

- **doubleKills :** Nombre de fois où le joueur tue 2 ennemis l'un après l'autre
- **dragonKills :** Nombre de fois où le joueur tue l'objects « dragon »
- **firstBloodAssist :** Si le joueur a contribué au premier mort de la partie
- **firstBloodKill :** Si le joueur est responsable du premier mort de la partie
- **firstTowerAssist :** Si le joueur a contribué à la destruction de la première tour de la partie
- **firstTowerKill :** Si le joueur est responsable de la destruction de la première tour de la partie
- **gameEndedInEarlySurrender :** Si le match a pris fin suite à un abandon en début de jeu
- **gameEndedInSurrender :** Si le match a pris fin suite à un abandon après 20 min
- **goldEarned :** Nombre d'or gagné durant la partie
- **goldSpent :** Nombre d'or dépensé durant la partie
- **individualPosition :** Identique à teamPosition
- **inhibitorKills :** Nombre de fois où le joueur à détruit un inhibiteur
- **inhibitorTakedowns :** Si le joueur a contribué à détruire un inhibiteur
- **inhibitorsLost :** Nombre de fois où l'équipe à perdu un inhibiteur
- **item0 :** L'objet / équipement acheté au slot 0
- **item1 :** L'objet / équipement acheté au slot 1
- **item2 :** L'objet / équipement acheté au slot 2
- **item3 :** L'objet / équipement acheté au slot 3
- **item4 :** L'objet / équipement acheté au slot 4
- **item5 :** L'objet / équipement acheté au slot 5
- **item6 :** L'objet / équipement acheté au slot 6
- **itemsPurchased :** Nombre d'équipements achetés
- **killingSprees :** Nombre de fois où le joueur à tueé beaucoup d'ennemis sans mourir
- **kills :** Nombre de fois où le joueur à tué un adversaire
- **lane :** Nom de la ligne sur laquelle le joueur a commencé sa partie
- **largestCriticalStrike :** Dégât critique le plus élevé, infligé à un ennemi
- **largestKillingSpree :** Nombre maximum d'adversaire tué sans mourir
- **largestMultiKill :** Nombre maximum d'adversaires tués à la suite, durant un court lapse de temps
- **longestTimeSpentLiving :** Temps maximum passé en vie (en ms)
- **magicDamageDealt :** Nombre de dégâts magiques appliquées
- **magicDamageDealtToChampions :** Nombre de dégâts magiques appliquées aux ennemis

-

magicDamageTaken : Nombre de dégâts magiques reçus

- **neutralMinionsKilled :** Nombre de monstres de la jungle tués
- **nexusKills :** Nombre de fois où le joueur a détruit le nexus adverse (0 ou 1)
- **nexusLost :** Si le jouer a perdu la partie
- **nexusTakedowns :** Si le joueur a contribué à la destruction du dernier objectif
- **objectivesStolen :** Nombre de fois où le joueurs à volé un objectif à l'ennemi
  objectivesStolenAssists : Nombre de fois où le joueur à contribué au vol d'un objectif ennemi
- **participantId :** Identifiant du joueur au sein de la partie
- **pentaKills :** Nombre de fois où le joueur élimine tous les joueurs ennemis à lui tout seul
- **physicalDamageDealt :** Nombre de dégâts physiques infligés
- **physicalDamageDealtToChampions :** Nombre de dégâts physiques infligés aux ennemis
- **physicalDamageTaken :** Nombre de dégâts physiques reçus
- **profileIcon :** Identifiant de l'image de profil du joueur
- **puuid :** Identifiant unique du joueur sur League Of Legend
- **quadraKills :** Nombre de fois où le joueur à effectué 4 kills d'affilé, dans un court l'apse de temps
- **riotIdName : -** - **riotIdTagline: -**
- **role :** Rôle du joueur dans la partie
- **sightWardsBoughtInGame :** Nombre de fois où le joueur a acheté une équipement de vision longue durée durant la partie
- **spell1Casts :** Nombre de fois où le joueur a envoyé son sort numéro 1
- **spell2Casts :** Nombre de fois où le joueur a envoyé son sort numéro 2
- **spell3Casts :** Nombre de fois où le joueur a envoyé son sort numéro 3
- **spell4Casts :** Nombre de fois où le joueur a envoyé son sort numéro 4
- **summoner1Casts :** Nombre de fois où le joueur à utilisé son sort d'invocateur 1
- **summoner1Id :** Identifiant du sort d'invocateur 1
- **summoner2Casts :** Nombre de fois où le joueur à utilisé son sort d'invocateur 2
- **summoner2Id :** Identifiant du sort d'invocateur 2
- **summonerId :** Identifiant unique de l'invocateur / joueur
- **summonerLevel :** Niveau general de l'invocateur / joueur
- **summonerName :** Nom du joueur en jeu
- **teamEarlySurrendered :** Si l'équipe du joueur s'est rendue dans les 15 premières minutes

-
- **teamId :** Identifiant de l'équipe dans la partie
- **teamPosition :** même que individualPosition
- **timeCCingOthers :** Nombre de fois où l'utilisateur à contrôlé la liberté de mouvement ou d'action d'un groupe de joueurs ennemis
- **timePlayed :** Temps joué au cours de la partie (en ms)
- **totalDamageDealt :** Nombre de dégâts infligés
- **totalDamageDealtToChampions :** Nombre de dégâts infligés aux ennemis
- **totalDamageShieldedOnTeammates :** Nombre de dégâts encaissé par un bouclier appliqué à un coéquipier
- **totalDamageTaken :** Nombre de dégâts encaissés
- **totalHeal :** Nombre de soins appliqués
- **totalHealsOnTeammates :** Nombre de soins appliqués aux alliés
- **totalMinionsKilled :** Nombre de sbires tués

  totalTimeCCDealt : Temps à entraver le mouvement des adversaires
- **totalTimeSpentDead :** Temps passé mort
- **totalUnitsHealed :** Nombre d'unités (sbire alliés, coéquipier) soigné
- **tripleKills :** Nombre de fois où le joueur a tué trois adversaires dans un cours lapse de temps
- **trueDamageDealt :** Nombre de dégâts pures infligés
- **trueDamageDealtToChampions :** Nombre de dégâts pures infligés aux ennemis
- **trueDamageTaken :** Nombre de dégâts pures encaissés
- **turretKills :** Nombre de tours détruites
- **turretTakedowns :** Nombre de tours pour lesquelles le joueur est responsable de la destruction, ou y a contribué
- **turretsLost :** Nombre de tours perdues par l'équipe du joueur
- **unrealKills :** Nombre de fois où le joueur à tué 6 ennemis sans mourir
- **visionScore :** Score de vision à la fin de la partie impliquant la vision fournie et annulée
- **visionWardsBoughtInGame :** Nombre d'objet de vision acheté au cours de la partie
- **wardsKilled :** Nombre d'objets de vision détruits
- **wardsPlaced :** Nombre d'objet de vision placé
- **win :** Si oui ou non le joueur a gagné