Projet d'analyse des données *Pokemon - Weedle's Cave*

Claire Guyot, Xinyu Wang, Zinan Zhou

12 juin 2019

1 Introduction

L'objectif de ce compte-rendu est de mettre en évidence les résultats des méthodes étudiées au cours du semestre appliquées à un jeu de données réelles (disponibles ici). Nous serons ainsi confrontées à un jeu de données qui peut présenter des manques ou aberrations.

Le jeu de données sur lequel nous allons travailler est composé de trois fichiers `pokemon.csv`, `combats.csv` et `tests.csv`, que nous allons décrire et analyser. Un aspect intéressant de l'analyse pourrait être la prédiction du gagnant d'un combat entre deux Pokémons en fonction des données d'entrée du combat.

Note : *Les termes du jeu de données sont en anglais, nous traduirons les mots qui ne sont pas transparents.*

2 Analyse exploratoire des données

2.1 Description des données

Le fichier `pokemon.csv` contient les caractéristiques de chaque Pokemon, `combats.csv` est composé de données de combats entre deux Pokémons, et `tests.csv` est un fichier permettant de tester et d'appliquer une méthode qui permet de prédire le résultat du combat. Ce dernier fichier ne nous interessera donc pas pour notre étude préalable des données.

Table 1 – Variables du jeu de données `pokemon.csv`

X.	id du Pokémon
Name	nom
Type.1	premier type d'attaque
Type.2	second type d'attaque
HP	points de vie
Attack	points d'attaque
Defense	points de défense
Sp.Atk	points d'attaque de l'attaque spéciale
Sp.Def	points de défense de l'attaque spéciale
Speed	vitesse d'attaque
Generation	état de développement
Legendary	statut légendaire

Table 2 – Variables du jeu de données `combats.csv`

First_pokemon	id du premier Pokémon
Second_pokemon	id du second Pokémon
Winner	id du Pokémon gagnant

En utilisant les fonctions `head(data)` et `sapply(data,class)`, il nous est possible d'avoir un bon aperçu des données (voir en annexe A, page 10). Nous pouvons constater que les classes des différentes variables correspondent aux données (par exemple, `Name` est bien un `factor`, `Attack` est bien un `integer`, et `Legendary` est bien un `factor` prenant les valeurs `True` et `False`).

La fonction `summary(data)` nous permet quant à elle de détecter les manques de données pour les différentes variables. Ainsi, nous remarquons qu'un Pokémon dont l'id est le 63 n'a pas de nom, ce qui peut être problématique pour l'identification de celui-ci. Cependant, ce Pokémon apparaît dans les données de combats et a gagné un grand nombre de combats. C'est pourquoi nous avons décidé de le garder au sein de notre jeu de données.

Aussi, sur les 800 Pokémons du jeu de données, 386 d'entre eux n'ont pas de deuxième type d'attaque, ce qui n'impacte pas de manière significative l'analyse des données. Nous pouvons également remarquer que seuls 65 parmi les 800 Pokémons sont légendaires.

2.2 Analyse exploratoire

Nous avons tout d'abord estimé intéressant de comparer les caractéristiques combatives des Pokémons légendaires et non légendaires.

Ainsi, les caractéristiques des Pokémons non légendaires (voir Fig. 1) sont globalement moins élévées que celles des Pokémons légendaires (voir Fig. 2). En moyenne, les caractéristiques des premiers sont de 69,54 contre 106,07 pour les seconds (voir Tab. 3).

L'écart le plus notoire concerne l'attaque et l'attaque spéciale, pour lesquelles on observe une différence de respectivement environ 40 et 55 points.

TABLE 3 – Comparaison des moyennes de chaque caractéristique entre Pokémons légendaires et non légendaires

	Non légendaires	**Légendaires**
HP	67.18	92.74
Attack	75.67	115.7
Defense	71.56	99.66
Sp.Atk	68.45	122.2
Sp.Def	68.89	105.9
Speed	65.46	100.2
	69.54	106.07

Par ailleurs, nous pouvons remarquer que les Pokémons légendaires ne monopolisent pas les caractéristiques les plus élevées. En effet, le maximum en `HP`, `Defense` et `Sp.Def` est atteint par des Pokémons non légendaires avec respectivement 255, 230 et 230 points contre un maximum de 150, 200 et 200 pour ceux légendaires.

En revanche, les caractéristiques les moins élevées sont atteintes par des Pokémons non légendaires exclusivement, pour toutes les caractéristiques de combats.

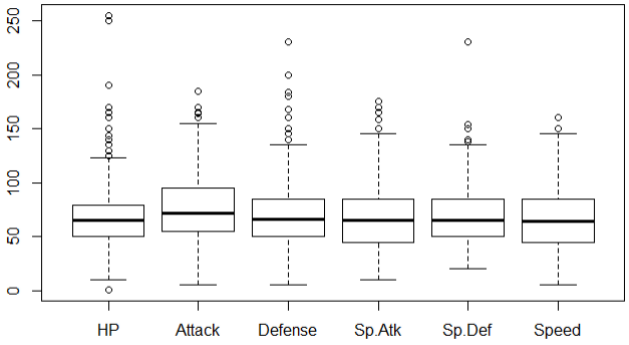

FIGURE 1 – Caractéristiques des Pokémons non légendaires

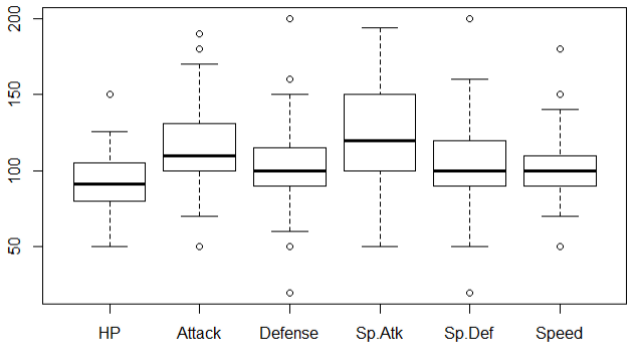

FIGURE 2 – Caractéristiques des Pokémons légendaires

Dans un premier temps, nous avons analysé la répartition des données selon les types (voir Fig.3).

Nous avons remarqué que la majorité des Pokémons étaient de type `Water`[1] (112 sur 800, soit 14%) et `Normal` (98 sur 800, soit 12,3%).

Certains autres types sont rares comme les types `Flying`[2] (4 sur 800, soit 0,5%) et `Fairy`[3] (17 sur 800, soit 2,1%).

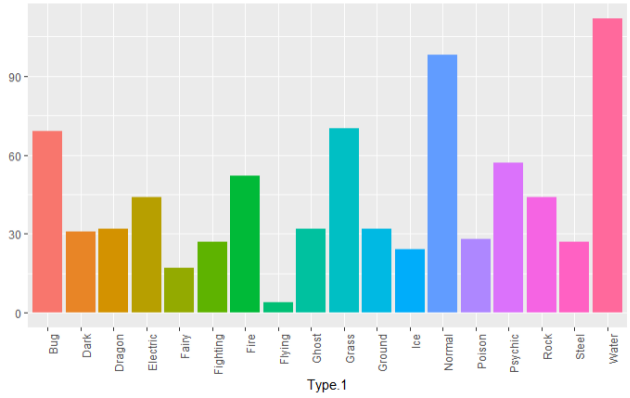

FIGURE 3 – Repartition des Pokémons selon les différents types

Parmi cette population, nous avons ensuite voulu estimé le nombre de Pokémons légendaires puisque ceux-ci ont globalement de meilleures caractéristiques et donc de meilleures chances de victoire *a priori* (voir Fig. 4).

La première chose que l'on peut remarquer est que certains types ne sont pas représentés parmi les Pokémons légendaires. C'est le cas des types `Bug`[4], `Fighting`[5] et `Poison`.

De plus, les types `Dragon` et `Psychic` sont largement plus représentés que tous les autres types avec respectivement 12 sur 65 (soit 18,5%) et 14 sur 65 (soit 21,5%).

Nous pourrons nous demander si cette différence de proportion a une incidence sur les résultats de combats, ou si la victoire est due à d'autres paramètres. En effet, elle pourrait également due à une différence d'attaque, de défense ou encore de vitesse par exemple.

Ainsi, les Pokémons de type `Dragon` et `Psychic` remportent-ils significativement plus facilement la victoire que d'autres types ?

Plus généralement, les Pokémons légendaires gagnent-ils plus que les Pokémons qui ne le sont pas ?

Nous tenterons d'apporter une réponse à ces questions par la suite.

1. Eau
2. Volant
3. Fée
4. Insecte
5. Combat

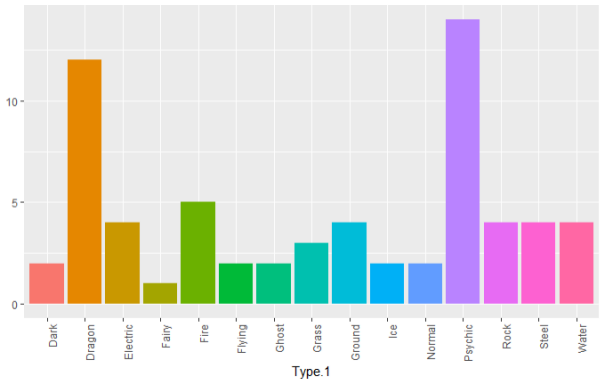

FIGURE 4 – Repartition des Pokémons légendaires selon les différents types

Dans une même optique, nous avons comparé les générations des différents Pokémons afin de pouvoir déterminer si d'une part, il existe une forte disparité entre les Pokémons, et d'autre part si une telle disparité a un impact sur les résultats des combats.

Sur la Fig. 5, nous pouvons observé qu'il n'y a pas de majeure différence de représentation des différentes générations de Pokémons, même si les générations 2, 4 et 6 sont légèrement moins représentées que les 1, 3 et 5 (envrion 100 contre 150 Pokémons pour chacune).

Ainsi, nous pouvons nous attendre à ce que les générations des Pokémons n'aient pas de conséquence sur la victoire ou la défaite lors d'un combat.

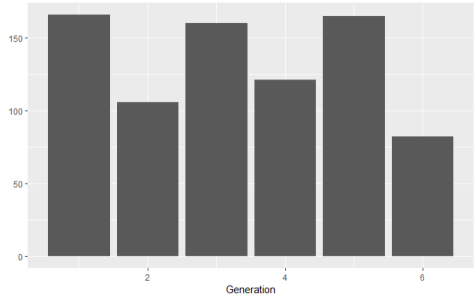

FIGURE 5 – Repartition des Pokémons selon les différentes générations

Dans la suite, afin d'avoir un meilleur aperçu de l'issue des combats et quels pourraient être les facteurs de victoire, nous avons regroupé dans un même `data.frame` les données de chaque Pokémon prenant part aux combats. Ainsi, nous avons facilité l'accès aux données pour les manipulations futures.

Nous avons donc ensuite étudié l'incidence de la génération du Pokémon sur la victoire ou la défaite.

Nous pouvons observer que la répartition des générations des vainqueurs est similaire à la répartition des générations dans la population totale des Pokémons (voir Fig. 6). Nous pouvons en déduire que la génération n'a pas d'impact sur la victoire d'un Pokémon.

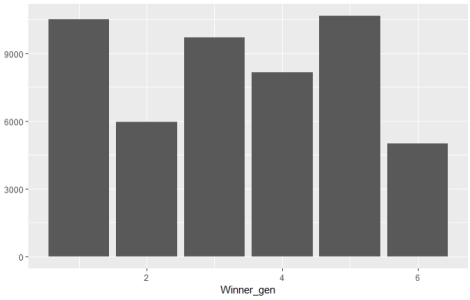

FIGURE 6 – Repartition des Pokémons vainqueurs selon leur génération

De la même manière, nous avons observé si le type d'un Pokémon ou le fait qu'un Pokémon soit légendaire a une incidence sur sa probabilité de victoire ou non.

Ainsi, sur la Fig. 7, nous remarquons que la répartition des types des Pokémons vainqueurs est presque identique à celle des types de la population totale de Pokémons dans notre jeu de données. Nous pouvons en déduire que le type d'un Pokémon a peu d'impact sur le résultat du combat.

De plus, nous avions vu que les Pokémons légendaires étaient principalement de type `Dragon` et `Psychic`. Nous pouvons voir sur cette même figure que que la proportion de victoire pour ces types est légèrement plus élevée que leur proportion dans la population de Pokémons. Nous pouvons en déduire que le fait d'être légendaire a une incidence, peu significative, sur les résultats des combats. Cette observation est due au fait que leurs caractéristiques de combat sont globalement plus élevées que celles de ceux non légendaires.

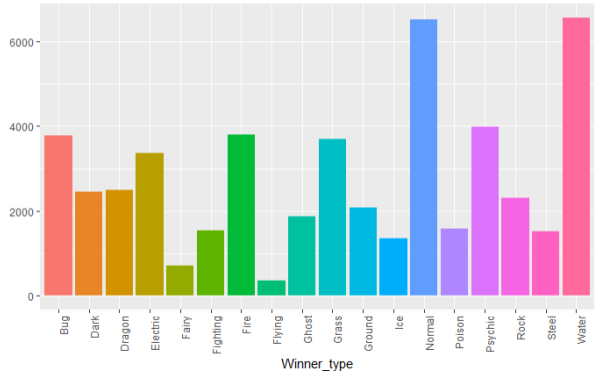

FIGURE 7 – Repartition des Pokémons vainqueurs selon leur type

Dans un dernier temps, nous avons transformé nos données initialement contenues dans le fichier `pokemon.csv` afin d'y ajouter le pourcentage de victoire pour chaque Pokémon, c'est-à-dire le nombre total de victoire sur le nombre total de combats menés par le Pokémon qu'il ait été en tant que `First_pokemon` ou `Second_pokemon`. De plus, nous avons transformé le booléen de la variable `Legendary` en un entier valant 0 si le booléen est faux et 1 sinon.

À partir de ces données, nous avons ensuite construit une matrice de corrélation avec les différentes variables numériques afin de déterminer l'incidence de chacune sur le pourcentage de victoire. À l'aide de la fonction `corrplot`, nous obtenons ainsi la visualisation de la matrice de corrélation.

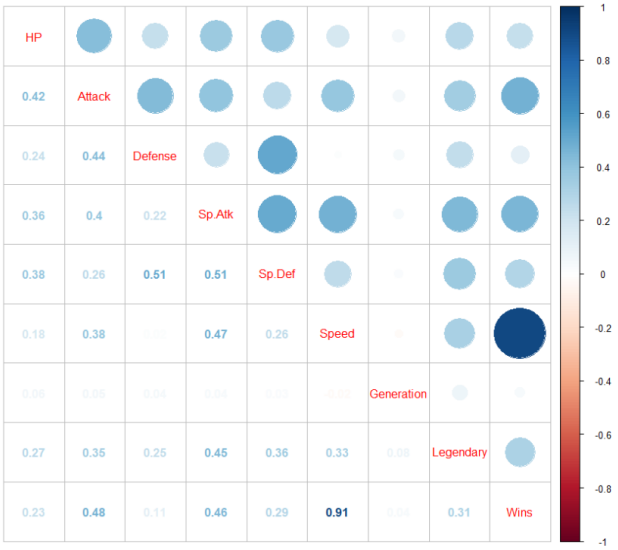

FIGURE 8 – Corrélation entre les différentes caractéristiques de combat et le pourcentage de victoire

D'après la Figure 8, nous pouvons observer que la vitesse est très corrélée au pourcentage de victoire avec une corrélation de 91%. Dans une moindre mesure, l'attaque est également corrélée au pourcentage de victoire avec 48% pour l'attaque et 46% pour l'attaque spéciale.

En revanche, nous pouvons voir que le nombre de points de vie, la défense, la défense spéciale, la génération, et le fait qu'un Pokémon soit légendaire ne sont que très peu corrélés au pourcentage de victoire, avec respectivement 23%, 11%, 29%, 4% et 31%.

En analysant les données de manière exploratoire, nous avons donc pu conclure sur les paramètres ayant une incidence sur la victoire d'un Pokémon. Ainsi, la vitesse d'attaque est la caractéristique de combat impactant le plus le victoire. L'attaque et l'attaque spéciale ont également un impact certain.

3 Méthodes non supervisées

3.1 Analyse en Composante Principale (ACP)

Nous avons mené une ACP sur notre jeu de données afin de déterminer l'importance de chaque composante principale, et de confirmer la corrélation entre les différentes caractéristiques de combat.

Ainsi, nous avons utilisé la fonction `princomp(data)` sur notre jeu de données préalablement centré et réduit, contenant les 6 caractéristiques de combat mesurables et également le pourcentage de victoire pour chaque Pokémon. Nous obtenons alors un histogramme indiquant l'importance de chacune des composantes principales (voir Fig. 9).

La premiere composante principale explique 45,7% de la variance, la deuxième en explique 20,3%, la troisième et la quatrième en explique respectivement 11,1% et 10,9%, et la cinquième en explique 6,9%. Pour expliquer plus de 90% de la variance, il faut donc les 5 premières composantes principales sur les 7 que nous avons. Cela signifie que les variables du jeu de données sont globalement indépendantes les unes des autres, bien que la première composante principale explique proche de 50% de la variance à elle seule.

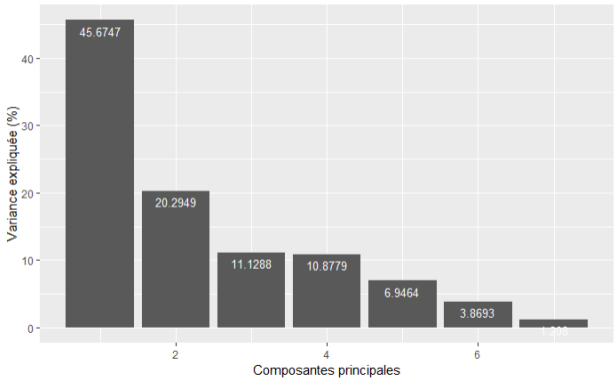

FIGURE 9 – Importance des différentes composantes principales

Nous représentons nos données sur le premier plan factoriel regroupant les première et deuxième composantes principales, qui à elles deux expliquent près de 70% de la variance (voir Fig. 10).

Nous pouvons observer que les données ne sont pas réparties en plusieurs groupes (ou *cluster*) mais forment plutôt un amas indifférencié.

À cela, nous ajoutons la représentation des vecteurs propres associés aux composantes principales. Ainsi, nous pouvons observer que l'attaque et l'attaque spé-

ciale sont fortement expliquées par la première composante principale, que la défense semble être expliquée plutôt par la deuxième composante, et que les points de vie et la défense spéciale semblent être expliquées par la première composante tout autant que la deuxième.

Aussi, nous pouvons confirmer la corrélation entre la vitesse d'attaque et le pourcentage de victoire des Pokémons puisque ces deux variables sont expliquées de la même manière dans le premier plan factoriel.

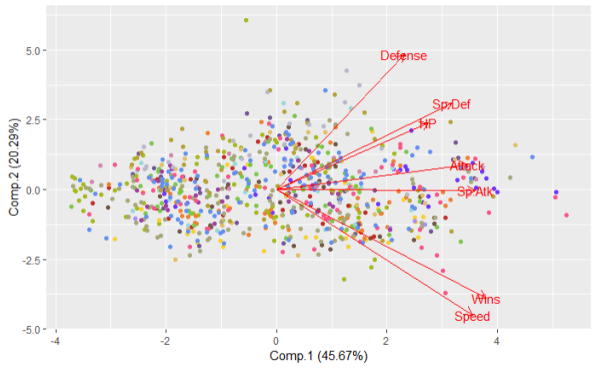

FIGURE 10 – Répartition des Pokémons dans le premier plan factoriel, colorés selon leur type, avec les vecteurs propres associés aux composantes principales

3.2 Méthode des K-means

La méthode des *k-means* est une méthode classification automatique permettant de regrouper les données en des *clusters*, c'est-à-dire des groupes d'individus partageant certaines caractéristiques communes.

En appliquant l'algorithme des *k-means* 100 fois pour des valeurs de k allant de 1 à 18 (le nombre de types différents), nous obtenons un histogramme montrant l'inertie intra-classe en fonction de k (voir Fig. 11).

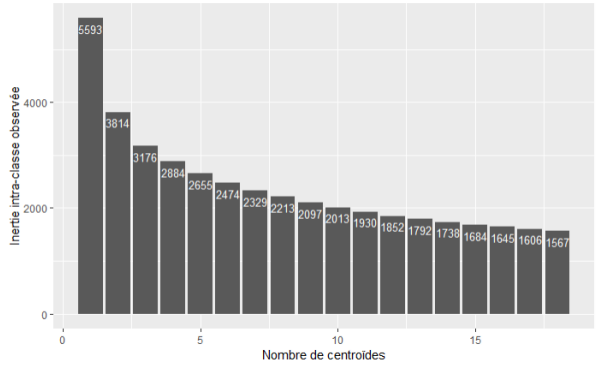

FIGURE 11 – Inertie intra-classe totale en fonction du nombres de centroïdes

Cet algorithme visant à minimiser l'inertie intra-classe converge, et permet d'obtenir une indication sur le nombre de classes optimal. Sur la figure ci-avant, nous pouvons observer que le nombre de *clusters* optimal est environ pour 3 centroïdes.

En utilisant le premier plan factoriel de l'ACP, nous pouvons ainsi représenter les *clusters* de notre jeu de données (voir Fig. 12).

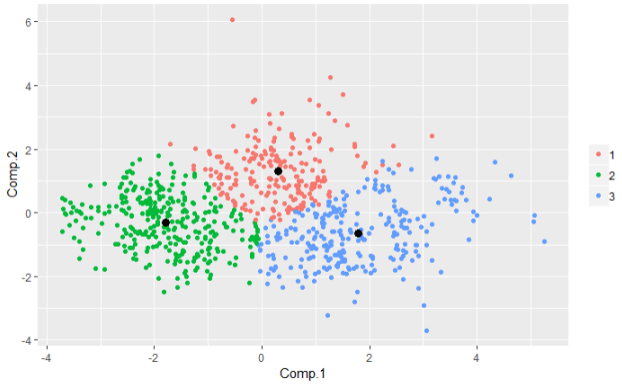

FIGURE 12 – *Clustering* des Pokémons dans le premier plan factoriel avec le méthode des *K-means* pour $k = 3$

Nous pouvons ensuite comparer les moyennes des différentes variables qui nous intéressent, afin de déterminer si le clustering est pertinent. Nous obtenons ainsi le tableau 4 ci-dessous.

Nous remarquons qu'il n'existe pas de différence notoire entre les différents *clusters*, et que rien ne semble permettre de pouvoir différencier si un individu appartient à tel ou tel *cluster*. La répartition en 3 classes ne semble donc pas pertinente pour notre jeu de données.

TABLE 4 – Moyennes des différentes variables selon les *clusters* identifiés par les *k-means*

Variable	Cluster 1	Cluster 2	Cluster 3
HP	70,39	67,63	69,73
Attack	79,63	79,31	77,66
Defense	73,81	73,16	74,8
Sp.Atk	73,19	71,36	74,21
Sp.Def	71,32	70,59	74,5
Speed	67,2	70,05	67,54
Wins	0,4879	0,4958	0,4888

Nous savons en revanche qu'il existe 18 types de Pokémons différents. Cette répartition en types est en quelque sort la classification existante de notre jeu de données. Sur la figure 10, nous pouvons voir que les 18 types ne forment pas des nuages de points distincts, et c'est pourquoi la méthode des *k-means* ne permet pas de les différencier.

4 Méthodes supervisées

Il nous a semblé intéressant pour la suite d'étudier la question *"Quel Pokémon est le gagnant dans un combat donné ?"*.

Cette question est d'autant plus pertinente que nous avons à notre disposition un ensemble d'apprentissage composé des 50000 données du fichier `combats.csv`. Nous pouvons créer un ensemble de test constitué de 10000 de ces données.

À chaque Pokémon du combat (1 pour *First_pokemon* ou 2 pour *Second_pokemon*), on associe le résultat prédit du combat (1 si le *First_pokemon* a gagné et 0 sinon).

4.1 Méthode des K plus proches voisins

Dans un premier temps, nous avons appliqué la méthodes des K plus proches voisins. Nous avons ainsi voulu déterminer si nous étions en mesure d'assigner une classe (*vainqueur* ou *perdant*) aux Pokémons de manière précise.

Pour ce faire, nous avons utilisé d'une part la méthode de calcul présente dans R appelée `knn`[6], et d'autre part une méthode de calcul implémentée à la main appelée `kppv`. Ainsi, nous avons calculé différents résultats (voir Tab. 5) que nous pouvons comparer afin de déterminer quelle méthode de calcul est la mieux adaptée dans notre cas.

Pour les différencier, nous avons utilisé différents indicateurs :

- *accuracy* : la proportion d'instances qui sont correctement classifiées
- *recall* : la proportion d'instances d'une classe qui sont correctement prédites
- *precision* : la proportion de prédictions correctes pour une classe
- *specificity* : la proportion de *perdants* qui sont correctement identifiés comme tels

Table 5 – Résultats de la classification par les *knn* et les *kppv*

Classification	Knn	Kppv
accuracy	0,756	0,715
recall	0,766	0,749
precision	0,726	0,675
specificity	0,748	0,686

De ces résultats, nous pouvons observer que nous avons un taux d'exactitude (*accuracy*) de plus de 70% avec les deux méthodes. Cela indique que près de 70% des Pokémons ont été correctement attribués à la classe *vainqueur* ou *perdant* selon leurs caractéristiques de combat et selon la classification présente dans l'ensemble d'apprentissage. Tous les autres indicateurs sont également autour de 70% pour chacune des méthodes.

6. *k nearest neighbors*

4.2 Analyse discriminante

Pour tenter de répondre à la question que nous nous sommes posée concernant la prédiction de l'issue d'un combat, nous avons mis en œuvre différentes méthodes d'analyse discriminante. Cette démarche nous semble pertinente puisque pour ces méthodes nous posons une hypothèse gaussienne des variables, et en effet les variables de notre jeu de données ont une distribution proche de la distribution gaussienne (voir Fig. 13).

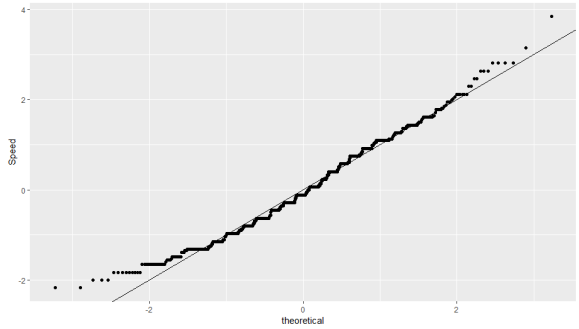

Figure 13 – Exemple de la distribution de la variable `Speed` par rapport à la distribution gaussienne

Par ces méthodes, nous avons ainsi obtenu différents résultats que nous avons comparé afin de déterminer la ou les méthodes les plus adaptées à nos données.

Il est tout d'abord utile de s'intéresser aux probabilités *a priori* de notre jeu de données. Nous avons pris la convention de nommer *1* le fait que le *First_pokemon* (c'est-à-dire le Pokémon qui attaque le premier) gagne, et *2* le fait que le *Second_pokemon* gagne. Ainsi, nous avons obtenu qu'*a priori*, le premier Pokémon gagne avec une probabilité d'environ 47%, et donc le second avec une probabilité d'environ 53%.

Table 6 – Probabilités *a priori*

	Calculées par R	Calculées
1	0,46678	0,47005
2	0,53323	0,52995

Pour chacune des méthodes d'analyse discriminante utilisées, nous avons utilisé d'une part la méthode présente au sein de `R` et d'autre part une méthode implémentée à la main.

4.2.1 Bayésien naïf

Pour la méthode bayésienne naïve, nous avons notamment utilisé la fonction `NaiveBayes(Xapp,zapp)` disponible à l'aide du package `klaR`.

Dans le tableau 7 ci-dessous, nous pouvons voir que la méthode native à R appelée *Nba* est plus efficace que notre méthode implémentée *Abn*, avec respectivement environ 89% et 70% d'exactitude.

TABLE 7 – Résultats de la classification bayésienne naïve

Classification	Nba	Abn
accuracy	0,885	0,699
recall	0,873	0,723
precision	0,881	0,666
specificity	0,895	0,677

Nous avons également calculé le taux d'erreur à partir du tableau de prédiction des données de test 8. *Nba* présente une erreur de 0,1149885 et *Abn* de 0,3009699.

TABLE 8 – Exactitude de la classification bayésienne naïve des données de test avec les deux méthodes

Nba	1	2
1	4110	554
2	596	4741

Abn	1	2
1	3404	1708
2	1302	3587

Certains résultats de cette méthode sont particulièrement intéressants (voir en annexe B.1, page 10). En effet, dans le tableau 9, nous pouvons observer que dans le cas de la variable `Speed` : si le premier Pokémon gagne, sa vitesse d'attaque est supérieure à celle du second Pokémon dans 95% des cas. Dans le cas contraire, la vitesse du second Pokémon est supérieure dans 88% des cas. La vitesse a donc une incidence sur la victoire.

TABLE 9 – Comparaison de la variable `Speed` selon le résultat prédit par la classification bayésienne naïve

Speed	1	2
1	0,047	0,953
2	0,878	0,122

4.2.2 Analyse discriminante linéaire

Pour l'analyse discriminant linéaire, nous avons notamment utilisé la fonction `lda(Xapp,zapp)` disponible à l'aide de la librairie `MASS`.

Dans le tableau 10 ci-dessous, nous pouvons voir que la méthode native à R appelée *Lda* et notre méthode implémentée *Adl* sont autant efficace avec un taux d'exactitude d'environ 0,70.

TABLE 10 – Résultats de l'analyse discriminante linéaire

Classification	Lda	Adl
accuracy	0,704	0,699
recall	0,650	0,717
precision	0,669	0,668
specificity	0,751	0,683

Nous avons également calculé le taux d'erreur à partir du tableau de prédiction des données de test 11. *Lda* présente une erreur de 0,295725 et *Adl* de 0,3005699.

TABLE 11 – Exactitude de l'analyse discriminante linéaire des données de test avec les deux méthodes

Lda	1	2
1	3061	1317
2	1645	3978

Adl	1	2
1	3376	1676
2	1330	3619

N'ayant que deux classes, *vainqueur* (valeur 1) et *perdant* (valeur 2), il nous a semblé plus pertinent de représenter la répartition des individus prédits selon ces deux valeurs.

Les erreurs de prédiction sont alors visibles, sur la figure 14, à travers le mélange de points bleus et oranges qui représentent respectivement les classes *vainqueur* et *perdant*.

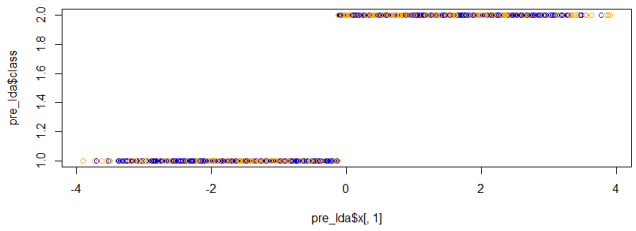

FIGURE 14 – Représentation de l'analyse discriminante linéaire

4.2.3 Analyse discriminante quadratique

Pour l'analyse discriminant quadratique, nous avons notamment utilisé la fonction `qda(Xapp,zapp)` disponible à l'aide de la librairie `MASS`.

Dans le tableau 12 ci-dessous, nous pouvons voir que la méthode native à R appelée *Qda* et notre méthode implémentée *Adq* sont autant efficace avec un taux d'exactitude d'environ 0,70.

TABLE 12 – Résultats de l'analyse discriminante quadratique

Classification	Qda	Adq
accuracy	0,704	0,699
recall	0,656	0,734
precision	0,698	0,662
specificity	0,747	0,667

Nous avons également calculé le taux d'erreur à partir du tableau de prédiction des données de test 13. *Lda* présente une erreur de 0,296025 et *Adq* de 0,3013699.

TABLE 13 – Exactitude de l'analyse discriminante quadratique des données de test avec les deux méthodes

Qda	1	2
1	3085	1337
2	1621	3958

Adq	1	2
1	3454	1762
2	1252	3533

Les erreurs de prédiction sont de nouveau visibles, sur la figure 15, à travers le mélange de points bleus et oranges qui représentent respectivement les classes *vainqueur* et *perdant*.

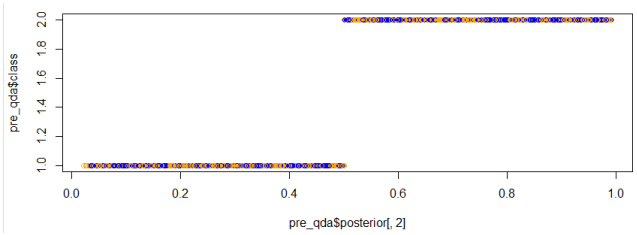

FIGURE 15 – Représentation de l'analyse discriminante quadratique

4.3 Régression logistique

Pour la régression logistique, nous avons notamment utilisé la fonction `glm(Xapp,zapp)` disponible à l'aide de la librairie `MASS`.

TABLE 14 – Résultats de la régression logistique

Classification	Reg.log
accuracy	0,885
recall	0,873
precision	0,881
specificity	0,895

Nous avons également calculé le taux d'erreur à partir du tableau de prédiction des données de test 15. *Reg.log* présente une erreur de 0,115.

TABLE 15 – Exactitude de la régression logistique des données de test

Winner	1	2
1	4110	554
2	596	4741

4.4 Arbres de décision

Pour les arbres binaires de décision, nous avons notamment utilisé la fonction `rpart(Xapp,zapp)` disponible à l'aide du package `rpart`.

Nous avons tout d'abord voulu déterminer l'arbre optimal pour notre jeu de données. Pour ce faire, il s'agit de récupérer l'erreur de validation croisée (*xerror*) pour chaque paramètre de complexité (*CP*) suite à la création d'un arbre binaire de décision avec nos données d'apprentissage.

L'arbre optimal est construit (voir Fig.16) en indiquant le paramètre de complexité associé à l'erreur de validation croisée minimale pour l'arbre d'apprentissage. Dans notre cas, cette valeur de *CP* est 8,8488e-06, pour une *xerror* de 0,18545 (voir Tab.16). Cet arbre optimal est construit en utilisant la variable `Speed`. Celle-ci est en effet celle qui détermine le plus la victoire.

TABLE 16 – Exactitude de la régression logistique des données de test

CP	***nsplit***	***rel error***	***xerror***	***xstd***
8.1455e-01	0	1.00000	1.00000	0.0053
3.1856e-05	1	0.18545	0.18545	0.0030
8.8488e-06	6	0.18529	0.18545	0.0030
0.0000e+00	12	0.18524	0.18551	0.0030

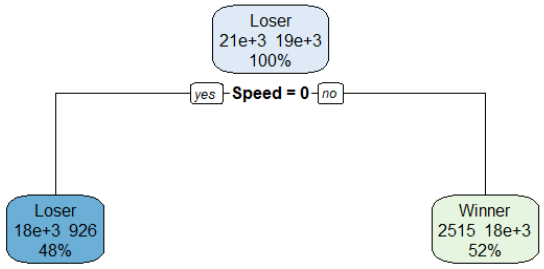

FIGURE 16 – Représentation de l'arbre binaire de décision optimal avec les données d'apprentissage

Nous avons ensuite appliqué cet arbre optimal à nos données de test, cet arbre est disponible en annexe B.2, page 11. Celui-ci a été construit en utilisant toutes les caractéristiques de combat.

Dans le tableau 17 ci-dessous, nous pouvons voir que la méthode native à R que nous appellerons *Tree.opt* a une exactitude de 92%.

TABLE 17 – Résultats de l'arbre binaire de décision optimal

Classification	Tree.opt
accuracy	0,917
recall	0,884
precision	0,956
specificity	0,954

Nous avons également calculé le taux d'erreur à partir du tableau de prédiction des données de test 18. *Tree* présente une erreur de 0,08289171.

TABLE 18 – Exactitude de l'arbre binaire de décision sur les données de test

Winner	1	2
1	4680	215
2	615	4491

4.5 Comparaison des différentes méthodes supervisées

Nous avons comparé l'erreur de chaque méthode supervisée, afin de déterminer laquelle est la mieux adaptée à notre jeu de données.

Nous avons tout d'abord remarqué que toutes les méthodes natives à `R` avaient une erreur plus petite que les méthodes implémentées à la main. Nous allons donc baser notre comparaison sur ces erreurs minimales.

Sur le tableau 19 ci-dessous, nous observons que la méthode d'arbre binaire de décision optimal à la taux d'erreur le plus réduit. Cette méthode est donc la plus efficace pour notre jeu de données.

Les méthodes bayésienne naïve et de régression logistique sont également pertinentes. Au contraire, les méthodes d'analyse discriminante linéaire et quadratique présentent le taux d'erreur le plus élevé.

En effet, notre jeu de données contient des variables qui ne suivent pas scrupuleusement une loi normale. Or, l'hypothèse gaussienne impacte grandement les analyses discriminantes. En outre, n'ayant que deux classe, la méthode de régression logistique est plus adaptée.

TABLE 19 – Comparaison des erreurs minimales

Knn	Nba	Lda	Qda	Reg.log	Tree.opt
0,244	0,115	0,296	0,296	0,115	0,083

5 Conclusion

Au cours de ce projet, nous avons pu mettre en évidence les résultats des méthodes étudiées tout au long du semestre appliquées à un jeu de données réelles. Ces données présentaient dans notre cas que peu de données manquantes ou incohérentes.

Nous avons été confrontées à un problème concret, à savoir : déterminer le paramètre garantissant la victoire d'un Pokémon lors d'un combat, et ainsi prédire l'issue de n'importe quel combat impliquant deux Pokémons.

Au travers de l'analyse exploratoire de nos données, nous avons pu mettre en évidence l'implication de la variable de vitesse d'attaque dans la victoire d'un Pokémon. Les différentes méthodes non supervisées nous ont permis de confirmer que ce paramètre *Speed* conditionnait la victoire.

Suite à cette confirmation, nous avons pu mettre en œuvre différentes méthodes supervisées de prédiction afin de déterminer les résultats des combats sur un ensemble de test à partir d'un ensemble d'apprentissage.

N'ayant que deux classes, *vainqueur* et *perdant*, certaines méthodes ont été plus difficiles à mettre en pratique comme l'analyse en composante principale ou l'analyse discriminante linéaire. De plus, les données étant réparties sur deux fichiers distincts, la mise en commun des données demande un traitement spécial.

Pour conclure, ce projet nous a permis de comprendre les différents aspects que nous avons vus durant le semestre. Nous avons mis en pratique les algorithmes des méthodes étudiées sur des données réelles, nous avons obtenus des résulats, et nous les avons interprétés pour en tirer des conclusion.

A Aperçu des données

TABLE 20 – Aperçu des données du fichier `pokemon.csv`

X.	Name	Type.1	Type.2	HP
1	Bulbasaur	Grass	Poison	45
2	Ivysaur	Grass	Poison	60
3	Venusaur	Grass	Poison	80
4	Mega Venusaur	Grass	Poison	80
5	Charmander	Fire	NA	39
6	Charmeleon	Fire	NA	58

Attack	Defense	Sp.Atk	Sp.Def
49	49	65	65
62	63	80	80
82	83	100	100
100	123	122	120
52	43	60	50
64	58	80	65

Speed	Generation	Legendary
45	1	False
60	1	False
80	1	False
80	1	False
65	1	False
80	1	False

TABLE 21 – Aperçu des données du fichier `combats.csv`

First_pokemon	Second_pokemon	Winner
266	298	298
702	701	701
191	668	668
237	683	683
151	231	151
657	752	657

TABLE 22 – Aperçu des données du fichier `tests.csv`

First_pokemon	Second_pokemon
129	117
660	211
706	115
195	618
27	656
126	222

B Résultats

B.1 Méthode bayésienne naïve

TABLE 23 – Comparaison des caractéristiques de combat selon le résultat prédit par la classification bayésienne naïve

HP	1	2
1	0,367	0,632
2	0,578	0,422

Attack	1	2
1	0,326	0,674
2	0,633	0,367

Def	1	2
1	0,429	0,571
2	0,527	0,473

Sp.Atk	1	2
1	0,330	0,670
2	0,619	0,381

Sp.Def	1	2
1	0,370	0,630
2	0,575	0,425

Speed	1	2
1	0,047	0,953
2	0,878	0,122

Legendary	0	1	2
1	0,027	0,851	0,122
2	0,114	0,855	0,031

B.2 Arbre binaire de décision

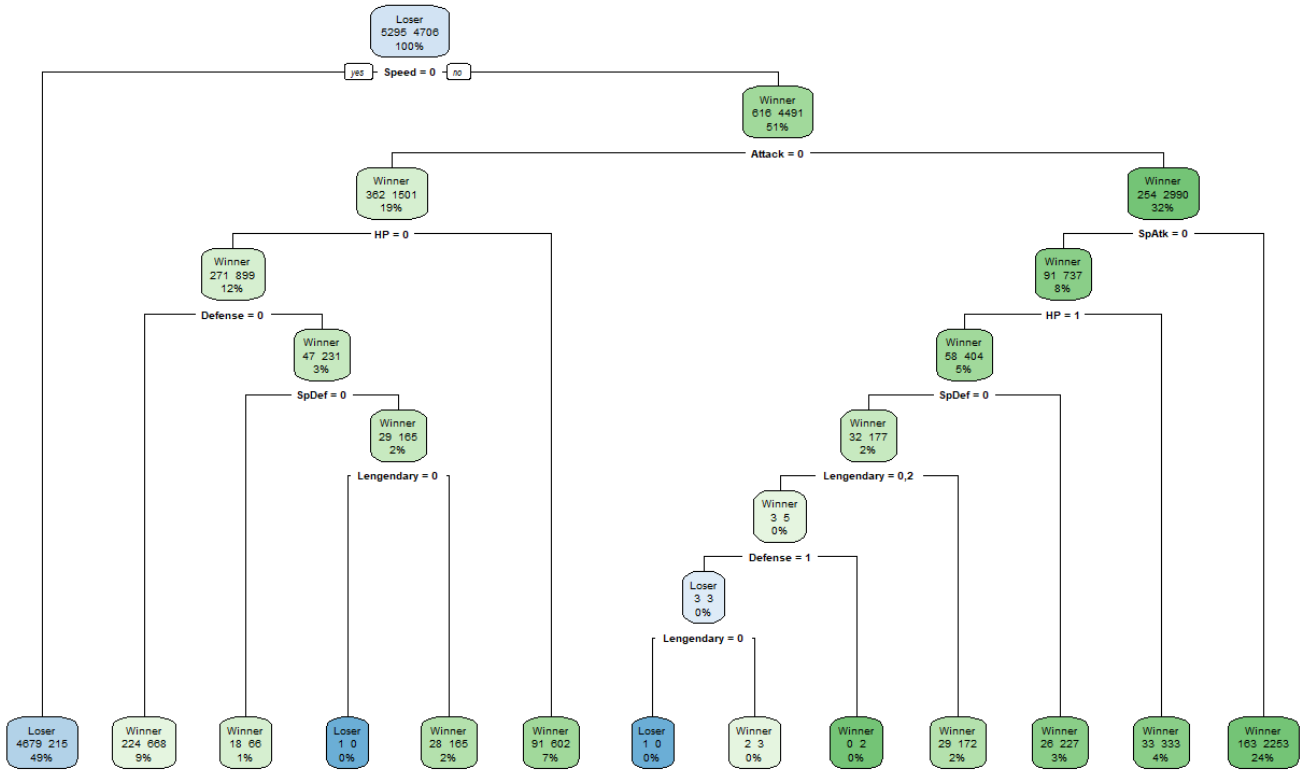

FIGURE 17 – Représentation de l'arbre binaire de décision optimal avec les données de test