



UNIVERSITÉ  
DE LORRAINE

PROJET LONG  
MASTER 2 INGÉNIERIE MATHÉMATIQUE  
POUR LA SCIENCE DES DONNÉES  
2022-2023

---

**Prédiction des résultats de Handball aux  
JO Paris 2024**

---

*Réalisé par :*  
Marcel MOUDILA  
Mouhamed FALL

*Enseignant :*  
Efoevi KOUDOU

En collaboration avec Florian FELICE, Christophe LEY  
Université de Luxembourg

18 octobre 2023

# Table des matières

<b>1 Généralités</b>	<b>2</b>
1.1 La science des données et le sport . . . . .	2
1.2 Brève présentation du handball . . . . .	2
<b>2 Analyse exploratoire des données</b>	<b>4</b>
2.1 Fichiers au format parquet et au format json . . . . .	4
2.2 Jeu des données coupe du monde handball masculin saison 2022-2023 . . . . .	5
2.2.1 Objectifs : . . . . .	5
<b>3 Présentation des méthodes</b>	<b>6</b>
3.1 Régression de Poisson . . . . .	6
3.1.1 cas du modèle de Poisson sur-dispersé . . . . .	6
3.1.2 cas du modèle de Poisson sous-dispersé . . . . .	7
3.2 Modèle d'approximation Gaussienne . . . . .	7
<b>4 Apprentissage et détermination du meilleur modèle</b>	<b>8</b>
4.1 Première approche . . . . .	8
4.2 Deuxième approche . . . . .	9
4.3 Troisième approche . . . . .	10
<b>5 Annexe</b>	<b>12</b>

# Chapitre 1

## Généralités

### 1.1 La science des données et le sport

Fort de son succès dans l'aide à la décision, la science des données inspire les chercheurs à collecter les données de différents sports (football, baseball, tennis, handball, basketball, etc) afin de les appliquer des algorithmes d'apprentissage automatique pour répondre à divers objectifs : évaluer la performance individuelle d'un athlète, prédire le vainqueur d'une compétition, etc.

Lors de l'évènement ultra-compétitif que sont les Jeux Olympiques, les athlètes recherchent tous les avantages qu'ils peuvent obtenir tout au long de leur préparation. Les entraîneurs commencent à collecter des données d'entraînement sur les jeunes athlètes, dans l'espoir de mieux comprendre l'origine des performances aux Jeux Olympiques et de déterminer les facteurs les plus déterminants.

Il suffit pour s'en convaincre de citer l'exemple du Paris Saint Germain qui a organisé un « Sports Analytics Challenge » en 2019. Le but de ce défi était de réunir des candidats issus des quatre coins du globe autour d'un projet big data. Les candidats avaient à plancher sur un jeu de données basé sur les matchs du championnat de France, prouvant que la science pouvait contribuer à améliorer les performances sportives.

Le but de notre projet est de prédire le vainqueur du tournoi de handball masculin aux Jeux Olympiques qui se tiendront à Paris en 2024. Pour cela, nous nous inspirons de l'état de l'art actuel, notamment l'article<sup>1</sup>

### 1.2 Brève présentation du handball

Le handball aux Jeux Olympiques comprend 12 équipes de 7 joueurs. Il y a une première phase de qualification et une phase à élimination directe (quart de finale, demi-finale et finale). Les équipes sont reparties en deux groupes de 6 équipes. les 4 meilleures équipes de chaque groupe poursuivent la compétition. Un match se déroule durant 2x30 minutes et l'équipe qui marque le plus de but à l'adversaire gagne le match. Un match nul est possible uniquement lors de la phase de qualification. L'arbitrage lors de la phase de qualification est fondée sur le nombre de points total obtenu par chaque équipe. Si deux équipes ont le même nombre total de points, d'autres critères sont pris en compte pour arbitrer de leurs sorts lors de la phase de qualification. nous renvoyons au site officiel [www.ihf.info](http://www.ihf.info) de la fédération internationale de handball pour plus de détails.

<sup>1</sup>. Groll, Andreas et al. 'Prediction of the 2019 IHF World Men's Handball Championship – A Sparse Gaussian Approximation Model'. 1 Jan. 2020 : 187 – 197.



FIGURE 1.1 – stade de handball

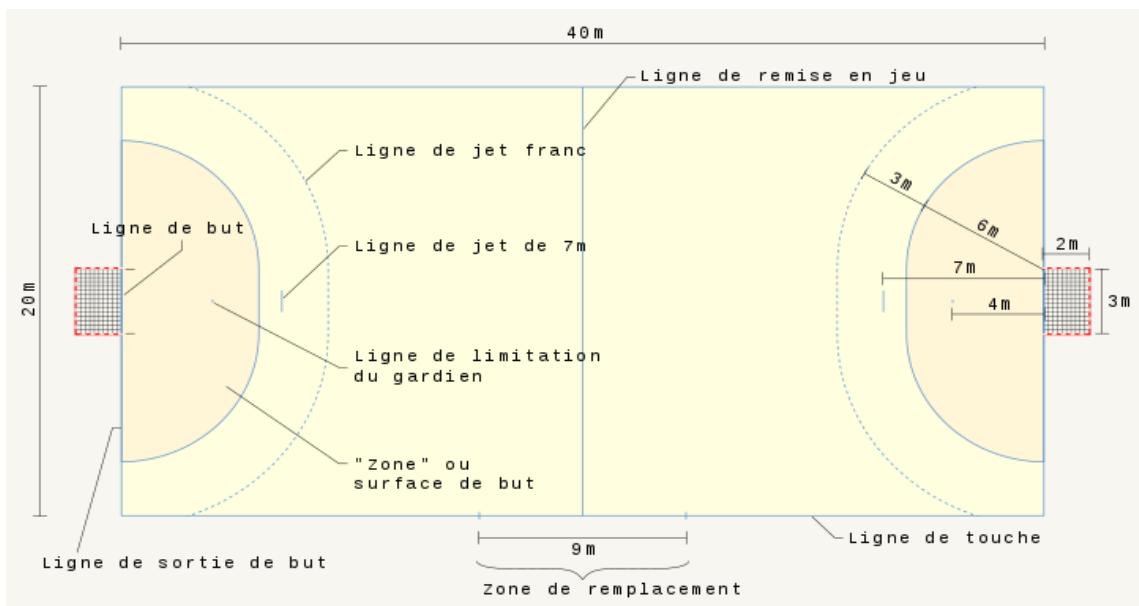


FIGURE 1.2 – différentes parties d'un terrain de handball

# Chapitre 2

## Analyse exploratoire des données

### 2.1 Fichiers au format parquet et au format json

En science des données, il existe de nombreux formats de fichiers pour stocker les données. Les plus basiques sont les fichiers csv, txt, et xlx qu'on peut ouvrir facilement avec la bibliothèque Pandas par exemple (Python). Quand on entre dans l'univers du big data, ces formats basiques sont peu recommandés en écriture comme en lecture, car l'espace de stockage alloué serait trop grand et la lecture ne serait pas très rapide. Pour les mêmes données, le format parquet prend moins d'espace de stockage, et est dix fois plus rapide en lecture qu'un format csv. On peut créer ou lire des fichiers parquet à partir de Python par exemple. Le format Json permet, entre autres fonctionnalités, de stocker des données textuelles de façon structurées.

Les données des matchs de handball nous ont été fournies en format parquet, nous les avons converties en format csv. Nous avons les données brutes des matchs (2997 observations, 19 variables), les données brutes des joueurs (1287 observations, 10 variables), et les données extraites des deux bases précédentes avec des variables calculées (528 observations, 44 variables). Ce sont les données avec des variables calculées que nous avons utilisées pour mener à bien notre projet.

1. name : nom des deux équipes (exemple France - Danemark)
2. start at : date et heure du match
3. home/away travel km : distance à vol d'oiseau effectuée par l'équipe home/away
4. home/away score final : nombre de buts marqué par l'équipe home/away
5. home/away nb players : nombre de joueurs de l'équipe home/away
6. home/away "caractéristique" "formule" : formule est soit la moyenne, soit l'écart-type ; caractéristique est, soit l'âge, soit la taille ou le poids des joueurs de l'équipe home/away. Exemple : away height avg (moyenne de la taille des joueurs de l'équipe away)
7. diff "poste" "caractéristique" "formule" : formule est soit la moyenne, soit l'écart-type ; caractéristique est soit la taille, soit l'âge ; poste est soit pivot, soit back, soit wing, soit gk. Exemple : diff pivot height avg (la différence de la taille moyenne des joueurs de l'équipe home au poste pivot et de la taille moyenne des joueurs de l'équipe away au poste pivot)
8. home/away team id : identifiant de l'équipe home/away
9. league id : identifiant de la compétition
10. season : année de la compétition
11. national team : équipe nationale
12. home/away ratio :

Nous avions aussi à notre disposition, un document Json, qui fournit, de façon détaillée, l'ensemble des compétitions, les saisons des compétitions, et qui précise également si la compétition est masculine ou féminine. Notre jeu des données avec les variables calculées contient que très peu de matchs de JO handball masculin (13 matchs) et beaucoup plus de matchs de coupe du monde handball masculin et seulement les matchs de la dernière saison de coupe du monde handball masculin (janvier 2023) sont en totalité.

		name	home_score_final	away_score_final
0		France - Denmark	29	34
1		Sweden - Spain	36	39
2		Egypt - Hungary	36	35
3		Germany - Norway	28	24
4		France - Sweden	31	26
5		Spain - Denmark	23	26
6		Norway - Hungary	33	25
7		Germany - Egypt	35	34

FIGURE 2.1 – Exemple des données des résultats de matchs

Nous avons donc travaillé avec le jeu des données des matchs de coupe du monde de handball masculin de la saison 2022-2023 (106 observations, 44 variables).

## 2.2 Jeu des données coupe du monde handball masculin saison 2022-2023

### 2.2.1 Objectifs :

Notre but est de prédire les résultats des matchs de demi-finales, de 3ème place, et de la finale.  
Nous disposons pour celà de trois approches.

1. prédire les buts marqués par chaque équipe (régression)
2. prédire la différence des buts marqués par les deux équipes (régression)
3. prédire la victoire ou la défaite d'une équipe (classification)

Nous avons donc crée deux nouvelles variables :

**diff score final** : qui pour un match France-Danemark est la différence des buts marqués par l'équipe France et l'équipe Danemark. Cette variable prend la valeur négative lorsque le nombre de but marqué par l'équipe France est inférieur à celui marqué par l'équipe Danemark.

**winner** : qui pour un match France-Danemark prend la valeur 1 si l'équipe de France gagne le match, la valeur -1 si l'équipe de France perd le match.

# Chapitre 3

## Présentation des méthodes

Dans ce chapitre , nous décrivons brièvement plusieurs approches de régression différentes qui entrent généralement en considération lorsque les buts marqués dans des matchs de handball simples sont directement modélisés. En fait, la plupart d'entre eux (ou de légères modifications de ceux-ci) ont déjà été utilisés dans d'anciennes recherches sur les données du football et, généralement, tous ont donné des résultats satisfaisants. Cependant, quelques ajustements sont nécessaires pour le handball. Toutes les méthodes décrites dans cette section peuvent être directement appliquées aux données . Par conséquent, chaque score est traité comme une seule observation et on obtient deux observations par match. Notre objectif est de choisir l'approche qui offre les meilleures performances en matière de prédiction, puis de l'utiliser pour prédire les JO de Paris 2024.

### 3.1 Régression de Poisson

Soit  $\lambda$  un paramètre et  $X$  une variable aléatoire réelle,  $X \sim P(\lambda)$  si quelque soit l'entier naturel  $k$ ,  
$$\mathbb{P}(X = k) = e^{-\lambda} \frac{\lambda^k}{k!}$$

Une approche traditionnelle qui est souvent appliquée, par exemple, pour modéliser les résultats du football est basée sur la régression de Poisson. Dans ce cas, les scores des équipes concurrentes sont traités comme des variables (conditionnellement) indépendantes suivant une distribution de Poisson (conditionnée sur certaines covariables). Comme déjà indiqué, chaque score d'un match de deux équipes de handball est traité comme une seule observation.

Un des inconvénients majeurs de la loi de poisson, est de vérifier l'hypothèse forte de la loi de Poisson  $\mathbb{E}[X] = \text{Var}(X) = \lambda$  . Ce qui est assez difficile à observer dans la réalité.

#### 3.1.1 cas du modèle de Poisson sur-dispersé

Dans le modèle de Poisson, la surdispersion se produit lorsque la variance est supérieure à la moyenne. Elle est considérée comme un problème car son omission peut entraîner une sous-estimation des écarts types des estimateurs, une variable peut apparaître à tort significative. La surdispersion est causée par la corrélation positive entre les variables ou par un excès de variation entre lesdites observations. Elle se produit également lorsque la distribution des données sont violées. On fait aussi référence à la surdispersion apparente, mais cette dernière se manifeste en cas de changements dans le modèle, comme une omission de variables explicatives importantes, une spécification non adéquate de la fonction de lien pour le modèle.

Lorsqu'une surdispersion est suspectée, il faut d'abord déterminer s'il s'agit d'une possibilité de surdispersion apparente, que l'on pourra corriger ou non, selon le cas qui se présente, en ajoutant des variables explicatives ou en utilisant la fonction de lien adéquate, etc. Toutefois, si la surdispersion persiste, différentes méthodes peuvent être employées, chacune se fondant sur la raison de ce problème. La détection de la sur-dispersion est facile à partir du rapport entre la déviance résiduelle et son degré de liberté. Si celui-ci vaut environ 1, on est dans le cas équi-dispersé, s'il est par contre supérieur à 1, on se trouve dans le cas sur-dispersé.

Pour prendre en compte la sur-dispersion des données, on introduit un paramètre de dispersion

noté  $\phi$  et un vecteur de poids a priori  $w$  pour obtenir la relation suivante :

$$V[Y] = \frac{\phi}{w} E[Y] = \frac{\phi}{w} \mu$$

Il en résulte que si  $\phi > 1$ , on a mise en évidence la sur-dispersion des données. Ainsi, on a généralisé le lien entre l'espérance et la variance car si on a  $\phi = 1$ , on se retrouve dans le modèle de Poisson habituelle.

On peut estimer le paramètre de dispersion par la façon suivante :

$$\hat{\phi} = \frac{1}{N - df} \sum_{i=1}^N r_i^2$$

où  $N$  désigne le nombre d'observation,  $df$  le nombre degrés de liberté et  $r_i$  les résidus de Pearson du modèle.

Parmi ces méthodes, nous présenterons la méthode de la régression binomiale négative.

### Modèle binomial négatif

La régression binomiale négative est utilisée pour modéliser les données de comptages pour lesquelles la variance est supérieure à la moyenne. Ce modèle est construit tel un modèle de mélange qui est utile pour ajuster la surdispersion de la distribution de Poisson. La vraisemblance du modèle binomial négatif est fondée sur le modèle de mélange Poisson-gamma. On distingue deux types de modèle de régression : NB1 (surdispersion constante) et NB2 (surdispersion variable).

Concrètement  $X$  suit une loi binomiale négative de paramètres  $r$  et  $p$ , ce que l'on représente aussi par :

$$X \sim BinNeg(r, p),$$

si sa fonction de masse est donnée par

$$\mathbb{P}(X = n) = \binom{n-1}{r-1} p^r (1-p)^{n-r}$$

pour chaque  $n = r, r+1, r+2, \dots$ . Son espérance et sa variance sont données respectivement par les formules suivantes :

$$\mathbb{E}[X] = \frac{r}{p} \quad \text{et} \quad \text{Var}(X) = \frac{r(1-p)}{p^2}$$

#### 3.1.2 cas du modèle de Poisson sous-dispersé

Il faut noter que dans certains cas que nous pouvons avoir la variance qui est inférieure à l'espérance de nos données, ou encore le coefficient de dispersion  $\phi < 1$ , on parlera alors de la sous-dispersion. Par exemple, il est possible d'avoir l'espérance des buts manqués soit supérieure à la variance. Toutefois, ce cas ne sera pas traité dans notre rapport.

## 3.2 Modèle d'approximation Gaussienne

Grâce au théorème de central limite, nous savons que pour de grandes valeurs de la moyenne de Poisson  $\lambda$ , la distribution de Poisson correspondante converge vers une distribution gaussienne (avec  $\mu = \sigma^2 = \lambda$ ). En pratique, cette approximation est faite lorsque  $\lambda \approx 30$  (ou plus).

Comme nous avons le nombre moyen de buts dans les matches de Coupe du monde de handball est proche de 30, cela nous a inspiré à appliquer également un modèle de réponse gaussien.

Cependant, au lieu de forcer la moyenne à égaler la variance, nous autorisons à ce que la variance diffère de la moyenne ( $\mu \neq \sigma^2$ ), c'est-à-dire une sur-dispersion ou une sous-dispersion.

Une variable aléatoire  $X$  suit une loi normale, c'est à dire  $X \sim \mathcal{N}(\mu, \sigma^2)$ , si sa densité est donnée par :

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

# Chapitre 4

## Apprentissage et détermination du meilleur modèle

Nous avons séparés les données en données d'apprentissage et en données de test. Les données d'apprentissage sont tous les matchs de la coupe du monde de handball masculin saison 2022-2023 sauf ceux des deux demi-finale, le match de 3ème place, et la finale. Les données de test sont ces quatres matchs cités. Nous cherchons le modèle qui est performant dans un premier temps pour les matchs de demi-finale, puis nous utilisons ce meilleur modèle pour prédire les résultats des matchs de la 3ème place, et de la finale.

### 4.1 Première approche

Ici, l'objectif est de prédire les buts marqués par chaque équipe. Nous avons mis en oeuvre trois modèles : la régression de Poisson, la régression binomiale négative, et le modèle d'approximation Gaussienne.

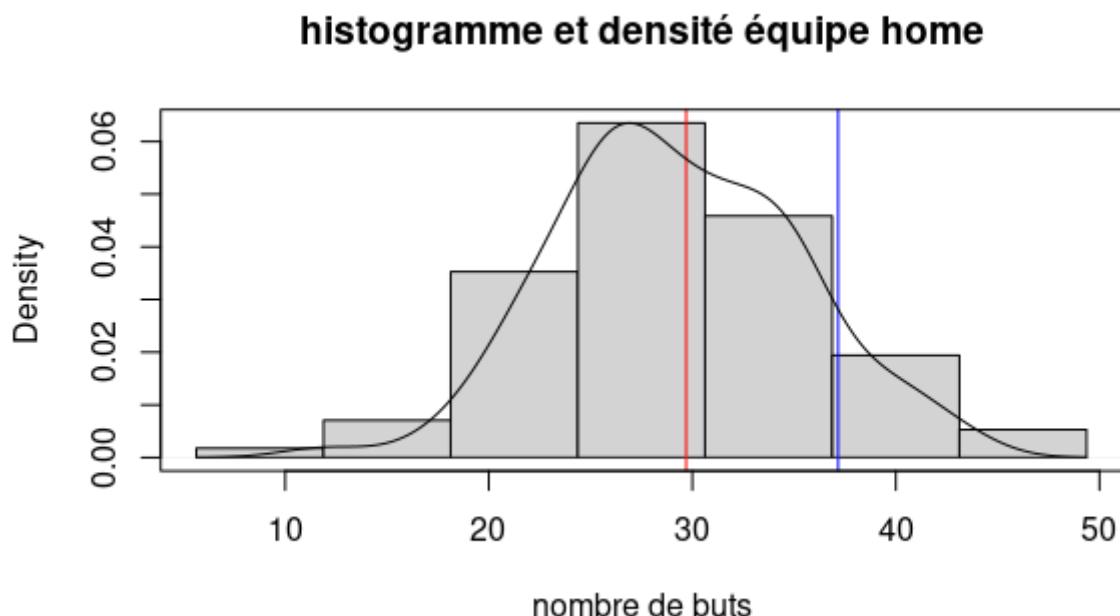


FIGURE 4.1 – moyenne des buts (en rouge), variance des buts (bleue)

Les trois modèles ne sont pas efficaces dans la prédiction des résultats des matchs de demi-finale de coupe du monde handball masculin session 2022-2023. Nous ne pouvons pas les utiliser pour

## histogramme et densité équipe away

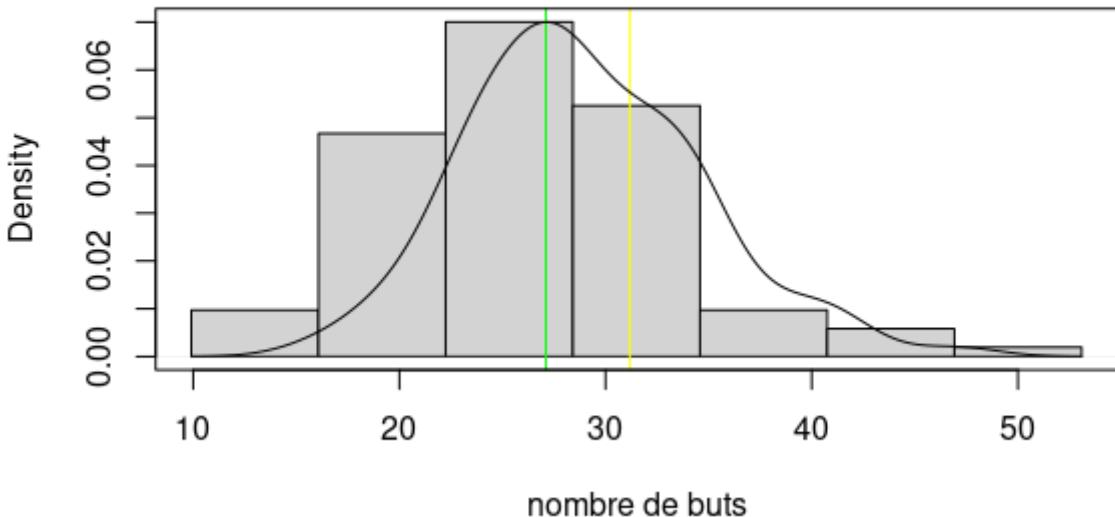


FIGURE 4.2 – moyenne des buts (en vert), variance des buts (en jaune)

modèle	RMSE	R2
Poisson	3.89	0.16
Binomiale négative	4.23	<b>0.33</b>
Approximation gaussienne	3.99	0.11

TABLE 4.1 – comparaison des modèles

la suite (c'est-à-dire la prédiction des matchs de 3ème place et la finale). On remarque néanmoins que le modèle binomiale négatif s'en tire mieux que les deux autres modèles car sa valeur de R2 est plus élevée. De plus, la variance des buts marqués est supérieure à la moyenne des buts marqués, donc, cas sur-dispersé, il est logique que le modèle binomial négatif soit plus performant que les deux autres modèles. Mais nous le recommandons pas car la valeur de R2 n'est que 0.33, ce qui signifie que seulement 33 % de la variabilité des données est représentée par le modèle.

## 4.2 Deuxième approche

Ici, l'objectif est de prédire la différence de buts marqués par les deux équipes. Nous avons mis en oeuvre deux méthodes d'ensemble : les forêts aléatoires de régression, et l'extrême gradient boosting (xgboost) de régression.

modèle	RMSE	R2
Forêts aléatoires	3.62	1
Xgboost	<b>3.11</b>	1

TABLE 4.2 – comparaison des modèles

Nous recommandons le modèle Xgboost car la valeur de RMSE est la plus petite. Nous remarquons que les valeurs de RMSE des deux modèles sont inférieures à celles des trois modèles de la première approche, donc la deuxième approche est beaucoup satisfaisante. Néanmoins, la prédiction faite pour le match de demi-finale Espagne-Danemark est une valeur positive, ce qui veut dire que le modèle prédit la victoire de l'équipe d'Espagne or ce résultat de match est faux car en réalité

l'Espagne a perdu ce match, nous voudrions plutôt une valeur négative comme prédiction. Du coup, nous apportons une troisième approche.

### 4.3 Troisième approche

Ici, l'objectif est de prédire la victoire ou la défaite d'une équipe. La variable cible est donc qualitative. Pour un match France-Danemark, qui n'est pas à élimination directe, la cible prend la valeur 1 si la France gagne le match, -1 si elle perd le match, et 0 si les scores de buts marqués sont égaux. Pour prédire les matchs de demi-finale, de 3ème place, et la finale, un score à égalité n'est pas possible.

Dans notre jeu d'entraînement, nous avons pour un match home-away 46 matchs où l'équipe home a perdu, 52 matchs où l'équipe home a gagné, et 3 matchs où l'équipe homme a marqué le même nombre de buts que l'équipe adverse.

Nous avons mis en œuvre deux modèles : les forêts aléatoires de classification, et la régression logistique multinomiale.

Nous avons pour les deux modèles, utilisé une cross-validation avec 5 folds. Ci-dessous les valeurs de l'accuracy pour les données d'apprentissage.

modèle	accuracy
Forêts aléatoires	<b>0.77</b>
Régression logistique	0.73

TABLE 4.3 – comparaison des modèles

Le modèle forêts aléatoires est plus performant que celui de la régression logistique sur les données d'apprentissage.

Ci-dessous les valeurs de l'accuracy sur les données de validation (les deux matchs de demi-finale)

modèle	accuracy
Forêts aléatoires	<b>1</b>
Régression logistique	0.5

TABLE 4.4 – comparaison des modèles

Le modèle forêts aléatoires est plus performant (encore) que celui de la régression logistique sur les données de validation. On remarque que le modèle de régression logistique est meilleur sur les données d'apprentissage et moins performant sur les données de validation. On remarque que le modèle des forêts aléatoires a prédit correctement les résultats (victoire ou défaite) des matchs de demi-finale. En effet, nous avons le tableau ci-dessous : Nous utilisons donc le modèle des forêts

match	vrai résultat	prédiction de l'équipe citée en premier
Espagne - Danemark	23-26	-1
France - Suède	31-26	1

TABLE 4.5 – prédiction des deux matchs de demi-finale

aléatoires pour prédire le vainqueur du match de la 3ème place, et du match de la finale. En voici, les résultats de l'accuracy sur ces nouvelles données de validation.

En effet, nous avons la table ci-dessous :

Notre meilleur modèle est donc le modèle des forêts aléatoires en utilisant l'approche de classification.

modèle	accuracy
Forêts aléatoires	<b>1</b>

TABLE 4.6 – accuracy lors de la prédiction pour la 3ème place, et pour la finale

match	vrai résultat	prédiction de l'équipe citée en premier
Suède - Espagne	36-39	-1
France - Danemark (finale)	29-34	-1

TABLE 4.7 – prédiction de la 3ème place et de la finale

### Conclusion :

Dans ce projet , nous avons d'abord utilisé sept modèles selon les objectifs de régression ou de classification et en considérant trois approches. Les données ont été les matchs de coupe du monde de handball masculin saison 2022-2023 à laquelle les nouvelles variables ont été créées pour affiner nos résultats. , en utilisant différentes approches pour les scores des matchs de handball en ce qui concerne leurs performances prédictives basées sur tous les matchs de Coupes du monde de handball masculin de saison 2022-2023.

Ces modèles sont : régression de Poisson, régression binomiale négative, régression Gaussienne, forêt aléatoire de régression, extrême gradient boosting de régression, forêt aléatoire de classification, et regression logistique multinomiale. La métrique accuracy a été utilisée pour l'approche de classification, et les métriques RMSE et R2 pour les deux approches de régression .

Si les deux approches de régression donnent des résultats peu satisfaisants sur nos données,l'approche de classification , quant à elle, donne pleine satisfaction, en prédisant exactement le vainqueur des matchs de demi-finale, de troisième place, et de finale.

Pour améliorer les résultats des deux approches de régression, on pourrait penser à plusieurs pistes : - utiliser les réseaux de neurones (en fournissant beaucoup plus de données d'entrainements), utiliser au moins trois saisons de coupe du monde de handball masculin pour les données d'entrainements.

### Remarques :

Les équipes pour les JO de handball masculin de Paris 2024 ne sont pas encore connues.

Chapitre **5**

## Annexe



# 28th IHF Men's World Championship 2023

Poland & Sweden



## Tournament Summary

Date	Start Time	Teams	Phase	Match No.	Half-time	Full time	1st Overtime	2nd Overtime	7m Shoot-out
THU 19 JAN	15:30	QAT - NED	Main Round - Group III	59	19 - 15	30 - 32			
	15:30	ALG - MKD	President's Cup Group II	60	11 - 19	25 - 40			
	15:30	USA - BRN	Main Round - Group IV	64	13 - 17	27 - 32			
	18:00	GER - ARG	Main Round - Group III	57	24 - 11	39 - 19			
	18:00	MAR - TUN	President's Cup Group II	61	13 - 13	25 - 30			
	18:00	EGY - BEL	Main Round - Group IV	62	22 - 15	33 - 28			
	20:30	NOR - SRB	Main Round - Group III	58	14 - 17	31 - 28			
	20:30	DEN - CRO	Main Round - Group IV	63	15 - 16	32 - 32			
FRI 20 JAN	15:30	SLO - ESP	Main Round - Group I	66	15 - 15	26 - 31			
	15:30	CHI - URU	President's Cup Group I	68	17 - 13	34 - 24			
	15:30	CPV - POR	Main Round - Group II	72	12 - 14	23 - 35			
	18:00	IRI - FRA	Main Round - Group I	67	14 - 18	29 - 41			
	18:00	KSA - KOR	President's Cup Group I	69	12 - 17	23 - 34			
	18:00	BRA - HUN	Main Round - Group II	70	14 - 14	25 - 28			
	20:30	MNE - POL	Main Round - Group I	65	8 - 11	20 - 27			
	20:30	ISL - SWE	Main Round - Group II	71	16 - 17	30 - 35			
SAT 21 JAN	15:30	SRB - ARG	Main Round - Group III	73	12 - 10	28 - 22			
	15:30	ALG - MAR	President's Cup Group II	76	15 - 13	27 - 28			
	15:30	BRN - EGY	Main Round - Group IV	79	9 - 13	22 - 26			
	18:00	QAT - NOR	Main Round - Group III	75	9 - 14	17 - 30			
	18:00	MKD - TUN	President's Cup Group II	77	14 - 16	28 - 33			
	18:00	CRO - BEL	Main Round - Group IV	78	21 - 13	34 - 26			
	20:30	NED - GER	Main Round - Group III	74	12 - 15	26 - 33			
	20:30	USA - DEN	Main Round - Group IV	80	10 - 18	24 - 33			
SUN 22 JAN	13:00	KOR - CHI	President's Cup Group I	84	16 - 16	26 - 33			
	15:30	MNE - SLO	Main Round - Group I	82	8 - 15	23 - 31			
	15:30	KSA - URU	President's Cup Group I	85	15 - 14	28 - 27			
	15:30	CPV - HUN	Main Round - Group II	88	15 - 22	30 - 42			
	18:00	IRI - POL	Main Round - Group I	83	10 - 16	22 - 26			
	18:00	BRA - ISL	Main Round - Group II	87	22 - 18	37 - 41			
	20:30	SWE - POR	Main Round - Group II	86	13 - 14	32 - 30			
	21:00	ESP - FRA	Main Round - Group I	81	13 - 13	26 - 28			
MON 23 JAN	15:30	QAT - ARG	Main Round - Group III	91	9 - 12	22 - 26			
	15:30	TUN - ALG	President's Cup Group II	92	15 - 12	30 - 25			
	15:30	USA - BEL	Main Round - Group IV	96	14 - 12	24 - 22			
	18:00	SRB - NED	Main Round - Group III	90	15 - 17	32 - 30			
	18:00	MKD - MAR	President's Cup Group II	93	18 - 12	40 - 25			
	18:00	CRO - BRN	Main Round - Group IV	95	17 - 16	43 - 32			
	20:30	GER - NOR	Main Round - Group III	89	16 - 18	26 - 28			
	20:30	EGY - DEN	Main Round - Group IV	94	12 - 17	25 - 30			
WED 25 JAN	13:00	URU - ALG	Placement Match 31/32	97	17 - 16	33 - 34			
	15:30	KSA - MAR	Placement Match 29/30	98	18 - 11	32 - 30			
	18:00	NOR - ESP	Quarterfinal	102	13 - 12	25 - 25	29 - 29	34 - 35	
	18:00	DEN - HUN	Quarterfinal	104	21 - 12	40 - 23			
	18:00	KOR - MKD	Placement Match 27/28	99	19 - 20	33 - 36			
	20:30	CHI - TUN	Placement Match 25/26	100	10 - 24	26 - 38			
	20:30	FRA - GER	Quarterfinal	101	16 - 16	35 - 28			
	20:30	SWE - EGY	Quarterfinal	103	14 - 9	26 - 22			
FRI 27 JAN	15:30	GER - EGY	Placement 5-8	106	17 - 14	30 - 30	35 - 34		
	18:00	ESP - DEN	Semifinal	105	10 - 15	23 - 26			
	18:00	NOR - HUN	Placement 5-8	107	16 - 13	33 - 25			
	21:00	FRA - SWE	Semifinal	108	16 - 12	31 - 26			
SUN 29 JAN	13:00	GER - NOR	Placement Match 5/6	109	16 - 13	28 - 24			
	15:30	EGY - HUN	Placement Match 7/8	110	17 - 11	28 - 28	31 - 31	36 - 35	
	18:00	SWE - ESP	Placement Match 3/4	111	22 - 18	36 - 39			
	21:00	FRA - DEN	Final	112	15 - 16	29 - 34			

Legend:									
Diff.	Goal Difference	GA	Goals Against	GF	Goals For				
L	Lost	MP	Matches Played	No.	Number				
Pts	Points	T	Tied	W	Won				

# Code R de notre projet long

MOUDILA Marcel - FALL Mouhamed

2023-03-27

```
# téléchargement des data sets #####
training <- read.csv("~/Documents/mes cours/projet long/V_TRAINING_INPUTS_INTERNATIONAL.csv", row.names=1)
games <- read.csv("~/Documents/mes cours/projet long/games_history.csv", row.names=1)
players <- read.csv("~/Documents/mes cours/projet long/players_club_history.csv", row.names=1)
# ouverture des tables #####
#View(training)
#View(games)
#View(players)
# handball masculin olympique 2020 #####
#data <- subset(training_2, league_id == c("7951"))
# le data training : reconnaissance des variables catégorielles #####
training$league_id <- as.factor(training$league_id)
training$season <- as.factor(training$season)
training$name <- as.factor(training$name)
training$home_team_id <- as.factor(training$home_team_id)
training$away_team_id <- as.factor(training$away_team_id)
training$national_team <- as.factor(training$national_team)
training$gender <- as.factor(training$gender)
# le data games : reconnaissance des variables catégorielles #####
games$league_id <- as.factor(games$league_id)
games$season <- as.factor(games$season)
games$name <- as.factor(games$name)
games$home_team_id <- as.factor(games$home_team_id)
games$away_team_id <- as.factor(games$away_team_id)
# effectif des matchs par league_id
table(training$league_id)

##
## 7940 7941 7951 7955 7956 7959 7960 7978
## 196   130    13    15    14     8   136    10
table(games$league_id)

##
## 7864 7918 7919 7920 7921 7922 7923 7924 7940 7941 7942 7943 7945
##    1   670    37     1   532    23    32    12   206   130     1     2   189
## 7946 7949 7951 7952 7954 7955 7956 7957 7959 7960 7961 7966 7970
##    2   102    38    30    12    18   140   105    87   226     52     2     9
## 7978 7984 8905 8944 9129 9583 10006 10088 10169
##   16    94    10     1    12    32     8    14    20

# création des nouvelles variables dans training
training <- transform(training, diff_score_final = home_score_final - away_score_final,
                      winner = ifelse(home_score_final > away_score_final, 1,
```

```

        ifelse(home_score_final == away_score_final, 0,-1)))
training$winner <- as.factor(training$winner)
#View(training)
# construction du data des matchs du championnat du monde #####
data <- subset(training, league_id==c("7940"))
#View(data)
# tri de ce data
data <- data[order(data$start_at, decreasing = F),]
#View(data)
# coupe du monde 2022-2023
data2022 <- subset(data, start_at >= 2023)
data2022$name <- as.numeric(data2022$name)
data2022$home_team_id <- as.numeric(data2022$home_team_id)
data2022$away_team_id <- as.numeric(data2022$away_team_id)
data2022$season <- as.numeric(data2022$season)
data2022$league_id <- as.numeric(data2022$league_id)
data2022$gender <- as.numeric(data2022$gender)
data2022$winner <- as.factor(data2022$winner)
data2022$national_team <- as.numeric(data2022$national_team)
data2022$home_nb_players <- as.numeric(data2022$home_nb_players)
data2022$away_nb_players <- as.numeric(data2022$away_nb_players)
#View(data2022)
# ensemble de test et ensemble d'apprentissage
test <- data2022[c(99,101),] # les deux demi-final
train <- data2022[-c(99,101,104,105),]
test <- subset(test, select = - c(name,start_at,season,gender,national_team))
train <- subset(train, select = - c(name,start_at, season,gender,national_team))
test2 <- data2022[c(104,105),] # classement et final
test2 <- subset(test2, select = - c(name,start_at,season,gender,national_team))
#View(train)
#View(test)
# séparation cible / covariable
X_train <- subset(train, select = -c(home_score_final,away_score_final,
                                         diff_score_final,winner))
Y_train <- subset(train, select = c(winner))
X_test <- subset(test, select = -c(home_score_final,away_score_final,
                                         diff_score_final,winner))
Y_test <- subset(test, select = c(winner))
X_test2 <- subset(test2, select = -c(home_score_final,away_score_final,
                                         diff_score_final,winner))
Y_test2 <- subset(test2, select = c(winner))
#View(X_train)
#View(X_test)
#View(Y_train)
#View(Y_test)
# enregistrement en csv
#write.csv2(X_train,file="Xtrain.csv")
#write.csv2(Y_train,file="Ytrain.csv")
#write.csv2(X_test,file="Xtest.csv")
#write.csv2(Y_test,file="Ytest.csv")
# entrainement, prédiction, et évaluation des modèles
library(caret)

## Le chargement a nécessité le package : ggplot2

```

```

## Le chargement a nécessité le package : lattice
# cross-validation avec 5 folds
fitControl <- trainControl(method = "cv", number = 5)
# random forest
randomForest <- train(x = X_train, y = Y_train$winner,
                      method = "rf",
                      trControl = fitControl,
                      metric = "Accuracy")
randomForest

## Random Forest
##
## 101 samples
## 37 predictor
##   3 classes: '-1', '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 81, 81, 80, 81, 81
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##     2    0.7728571  0.5567936
##    19    0.7533333  0.5193159
##    37    0.7233333  0.4665372
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
pred_randomForest <- predict(randomForest, X_test)
pred_randomForest

## [1] -1 1
## Levels: -1 0 1
postResample(pred = pred_randomForest, obs = Y_test$winner)

## Accuracy   Kappa
##      1       1
# match de classement et final
pred_randomForest2 <- predict(randomForest, X_test2)
pred_randomForest2

## [1] -1 -1
## Levels: -1 0 1
postResample(pred = pred_randomForest2, obs = Y_test2$winner)

## Accuracy   Kappa
##      1       NA
# régression logistique multinomiale
regLogistique <- train(X_train, Y_train$winner,
                      method = "multinom",
                      trControl = fitControl,
                      metric = "Accuracy",

```

```

          trace = FALSE)
regLogistique

## Penalized Multinomial Regression
##
## 101 samples
## 37 predictor
## 3 classes: '-1', '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 80, 81, 80, 81, 82
## Resampling results across tuning parameters:
##
##   decay  Accuracy  Kappa
##   0e+00  0.6925063  0.4475852
##   1e-04  0.7020301  0.4660714
##   1e-01  0.7325564  0.4893799
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.

pred_regLogistique <- predict(regLogistique, X_test)
pred_regLogistique

## [1] 1 1
## Levels: -1 0 1
postResample(pred = pred_regLogistique, obs = Y_test$winner)

## Accuracy  Kappa
##      0.5      0.0

# deuxième approche
Y_train <- subset(train, select = c(diff_score_final))
Y_test <- subset(test, select = c(diff_score_final))
#plot(Y_train$diff_score_final)
#View(Y_train)
#View(Y_test)
# random forest
randomForest3 <- train(x = X_train, y = Y_train$diff_score_final,
                        method = "rf",
                        trControl = fitControl,
                        metric = "RMSE")
randomForest3

## Random Forest
##
## 101 samples
## 37 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 80, 81, 82, 81, 80
## Resampling results across tuning parameters:
##

```

```

##    mtry   RMSE    Rsquared    MAE
##    2     7.207823 0.4816754 5.479763
##    19    6.581320 0.5168909 5.050103
##    37    6.765041 0.4759995 5.285244
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 19.
pred_randomForest3 <- predict(randomForest3, X_test)
pred_randomForest3

##      5      4
## 1.2730 2.1772

postResample(pred = pred_randomForest3, obs = Y_test$diff_score_final)

##      RMSE Rsquared    MAE
## 3.621238 1.000000 3.547900

# xgbLinear
library(xgboost) # (à modifier)
xgb <- train(x = X_train, y = Y_train$diff_score_final,
              method = "xgbLinear",
              trControl = fitControl,
              metric = "RMSE")
xgb

## eXtreme Gradient Boosting
##
## 101 samples
## 37 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 81, 80, 81, 81, 81
## Resampling results across tuning parameters:
##
##    lambda  alpha  nrounds   RMSE    Rsquared    MAE
##    0e+00  0e+00    50     8.144465  0.3933869  6.539944
##    0e+00  0e+00   100     8.144465  0.3933870  6.539944
##    0e+00  0e+00   150     8.144465  0.3933870  6.539944
##    0e+00  1e-04    50     8.167993  0.3906183  6.546513
##    0e+00  1e-04   100     8.167993  0.3906183  6.546512
##    0e+00  1e-04   150     8.167993  0.3906183  6.546512
##    0e+00  1e-01    50     8.156995  0.3749530  6.499049
##    0e+00  1e-01   100     8.156995  0.3749530  6.499049
##    0e+00  1e-01   150     8.156995  0.3749530  6.499049
##    1e-04  0e+00    50     8.168053  0.3905044  6.548314
##    1e-04  0e+00   100     8.168053  0.3905044  6.548313
##    1e-04  0e+00   150     8.168053  0.3905044  6.548313
##    1e-04  1e-04    50     8.172643  0.3899671  6.553648
##    1e-04  1e-04   100     8.172643  0.3899671  6.553648
##    1e-04  1e-04   150     8.172643  0.3899671  6.553648
##    1e-04  1e-01    50     8.161049  0.3742280  6.505054
##    1e-04  1e-01   100     8.161049  0.3742280  6.505054
##    1e-04  1e-01   150     8.161049  0.3742280  6.505054

```

```

##   1e-01  0e+00  50      8.114385  0.3646053  6.441991
##   1e-01  0e+00  100     8.114384  0.3646053  6.441991
##   1e-01  0e+00  150     8.114384  0.3646053  6.441991
##   1e-01  1e-04  50      8.117082  0.3642927  6.445355
##   1e-01  1e-04  100     8.117082  0.3642927  6.445355
##   1e-01  1e-04  150     8.117082  0.3642927  6.445355
##   1e-01  1e-01  50      8.058007  0.3507483  6.361820
##   1e-01  1e-01  100     8.058007  0.3507483  6.361820
##   1e-01  1e-01  150     8.058007  0.3507483  6.361820
##
## Tuning parameter 'eta' was held constant at a value of 0.3
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 50, lambda = 0.1, alpha
## = 0.1 and eta = 0.3.

pred_xgb <- predict(xgb, X_test)
pred_xgb

## [1] 0.9782017 3.1081340
postResample(pred = pred_xgb, obs = Y_test$diff_score_final)

##      RMSE Rsquared      MAE
## 3.114903 1.000000 2.935034

# première approche
Y_train_home <- subset(train, select = c(home_score_final))
Y_test_home <- subset(test, select = c(home_score_final))
Y_train_away <- subset(train, select = c(away_score_final))
Y_test_away <- subset(test, select = c(away_score_final))
# moyenne et variance de Y_train_home
moyenne_Y_train_home <- mean(Y_train_home$home_score_final)
moyenne_Y_train_home

## [1] 29.25743
variance_Y_train_home <- var(Y_train_home$home_score_final)
variance_Y_train_home

## [1] 35.23307

# moyenne et variance de Y_train_away
moyenne_Y_train_away <- mean(Y_train_away$away_score_final)
moyenne_Y_train_away

## [1] 28.94059
variance_Y_train_away <- var(Y_train_away$away_score_final)
variance_Y_train_away

## [1] 32.23644

# regression de Poisson
library(glmnet)

## Le chargement a nécessité le package : Matrix
## Loaded glmnet 4.1-6

```

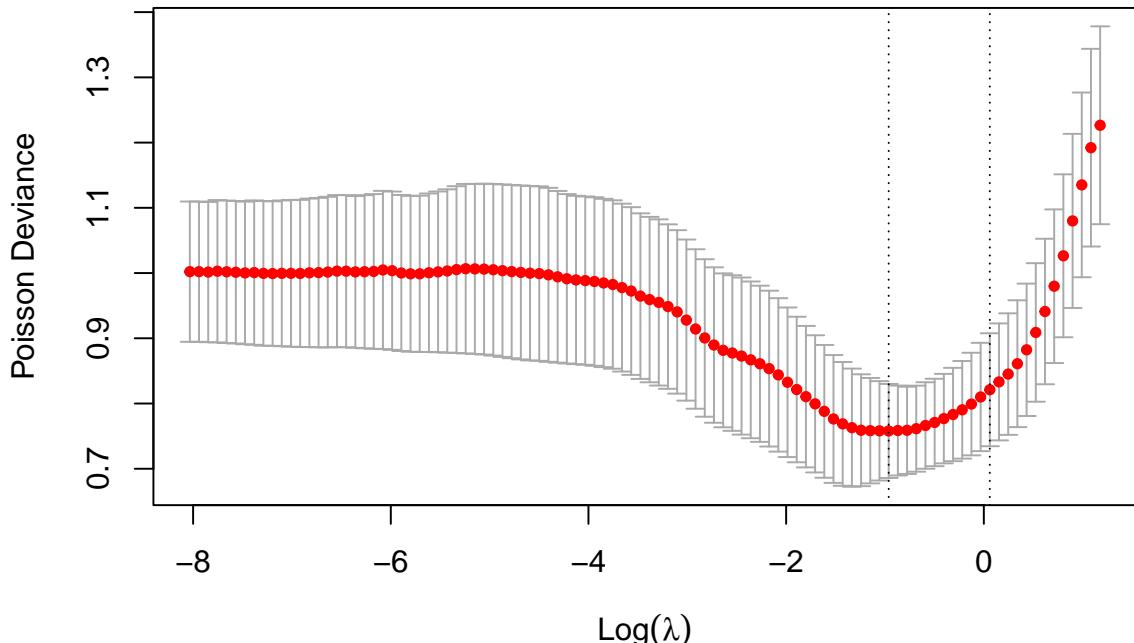
```

# home
model_Poisson_home <- cv.glmnet(x = as.matrix(X_train), y = Y_train_home$home_score_final,
                                   type.measure = c("deviance"),
                                   family = c("poisson"))
model_Poisson_home

##
## Call: cv.glmnet(x = as.matrix(X_train), y = Y_train_home$home_score_final,      type.measure = c("d
##
## Measure: Poisson Deviance
##
##      Lambda Index Measure      SE Nonzero
## min 0.3819    24  0.7579 0.07207      14
## 1se 1.0626    13  0.8211 0.08660      3
plot(model_Poisson_home)

```

35 35 35 34 33 32 30 30 26 22 17 13 8 3 2 1



```

pred_model_Poisson_home <- predict(model_Poisson_home, as.matrix(X_test), interval="pred")
pred_home <- exp(pred_model_Poisson_home)
# away
model_Poisson_away <- cv.glmnet(x = as.matrix(X_train), y = Y_train_away$away_score_final,
                                   type.measure = c("deviance"),
                                   family = c("poisson"))
model_Poisson_away

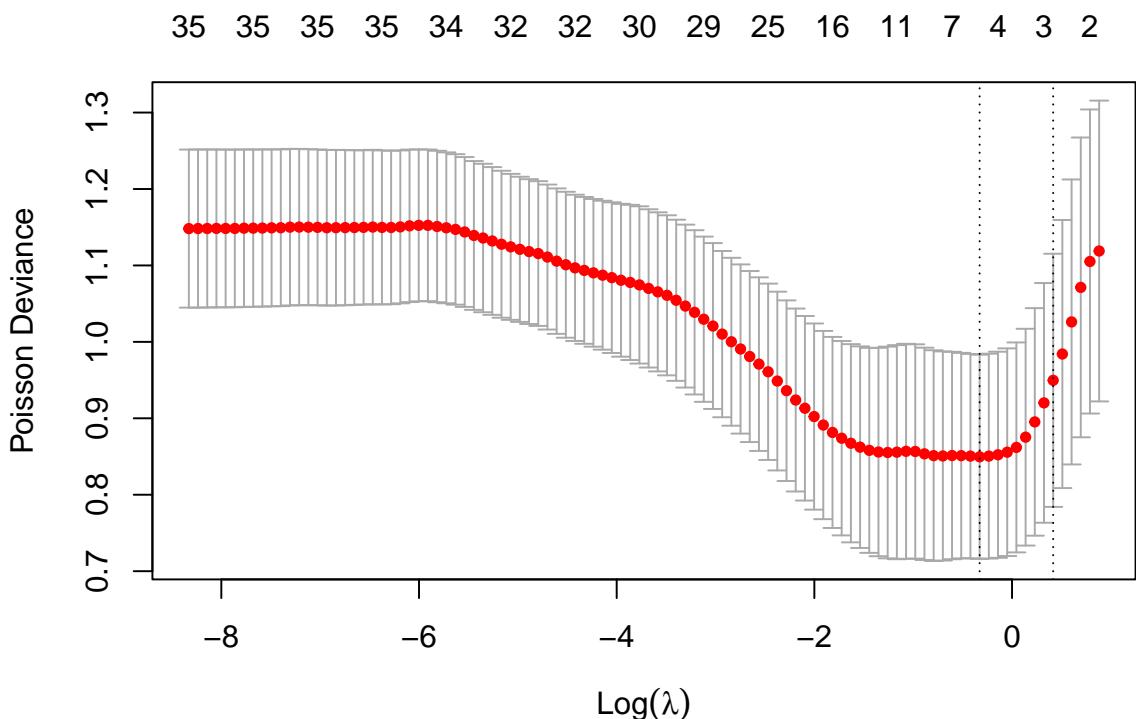
##
## Call: cv.glmnet(x = as.matrix(X_train), y = Y_train_away$away_score_final,      type.measure = c("d
##
## Measure: Poisson Deviance
##
##      Lambda Index Measure      SE Nonzero
## min 0.3819    24  0.7579 0.07207      14
## 1se 1.0626    13  0.8211 0.08660      3

```

```

## min 0.7205    14  0.8496  0.1336      6
## 1se 1.5165     6  0.9497  0.1656      3
plot(model_Poisson_away)

```



```

pred_model_Poisson_away <- predict(model_Poisson_away, as.matrix(X_test), interval="pred")
pred_away <- exp(pred_model_Poisson_away)
# validation
true_home <- Y_test_home$home_score_final
true_away <- Y_test_away$away_score_final
pred <- c(pred_home[1], pred_away[1], pred_home[2], pred_away[2])
obs <- c(true_home[1], true_away[1], true_home[2], true_away[2])
postResample(pred = pred,
             obs = obs)

##      RMSE Rsquared      MAE
## 3.8962293 0.1603112 3.2552631

# régression binomiale négative car moy < var
library(MASS)
# home
train_NB_home <- cbind(Y_train_home, X_train)
model_NB_home <- glm(home_score_final ~ ., data = train_NB_home,
                      family = negative.binomial(2))
summary(model_NB_home)

##
## Call:
## glm(formula = home_score_final ~ ., family = negative.binomial(2),
##      data = train_NB_home)
##
## Deviance Residuals:

```

```

##      Min       1Q     Median      3Q      Max
## -0.51542 -0.10209 -0.02157  0.09787  0.39587
##
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)          6.899e+00 2.914e+00  2.367  0.02090 *
## home_team_id        -1.224e-02 3.860e-03 -3.172  0.00231 **
## away_team_id         8.111e-03 3.970e-03  2.043  0.04512 *
## league_id            NA          NA          NA          NA
## home_travel_km       NA          NA          NA          NA
## away_travel_km      -1.097e-05 5.759e-06 -1.905  0.06127 .
## home_nb_players     -1.627e-02 1.930e-02 -0.843  0.40223
## home_clubs_ratio   -2.837e-01 2.081e-01 -1.364  0.17735
## home_age_avg        4.257e-01 2.668e-01  1.596  0.11542
## home_age_std        2.216e-01 2.747e-01  0.807  0.42279
## home_height_avg    3.702e-02 1.369e-01  0.270  0.78770
## home_height_std    -3.179e-01 2.230e-01 -1.425  0.15883
## home_weight_avg    -4.107e-02 2.859e-02 -1.436  0.15566
## home_weight_std    1.192e-02 1.807e-02  0.660  0.51161
## away_nb_players    -6.127e-03 1.969e-02 -0.311  0.75666
## away_clubs_ratio   1.501e-01 2.050e-01  0.732  0.46663
## away_age_avg       -4.580e-01 2.733e-01 -1.676  0.09854 .
## away_age_std       -2.106e-01 2.718e-01 -0.775  0.44117
## away_height_avg   -4.949e-02 1.338e-01 -0.370  0.71275
## away_height_std   2.887e-01 2.242e-01  1.287  0.20250
## away_weight_avg   5.213e-02 3.034e-02  1.718  0.09050 .
## away_weight_std   -3.323e-02 1.848e-02 -1.799  0.07673 .
## diff_wing_height_avg -4.088e-02 7.088e-02 -0.577  0.56610
## diff_back_height_avg 1.493e-02 7.359e-02  0.203  0.83982
## diff_pivot_height_avg 3.331e-02 3.867e-02  0.861  0.39224
## diff_gk_height_avg -1.823e-02 1.627e-02 -1.121  0.26659
## diff_wing_height_std 1.289e-01 8.649e-02  1.490  0.14095
## diff_back_height_std 1.368e-01 9.814e-02  1.394  0.16818
## diff_pivot_height_std 9.179e-02 5.668e-02  1.619  0.11019
## diff_gk_height_std -7.184e-02 2.334e-02 -3.077  0.00306 **
## diff_wing_age_avg -9.080e-02 5.534e-02 -1.641  0.10566
## diff_back_age_avg -1.361e-01 9.058e-02 -1.503  0.13773
## diff_pivot_age_avg -8.346e-02 4.852e-02 -1.720  0.09020 .
## diff_gk_age_avg   -8.288e-02 5.776e-02 -1.435  0.15610
## diff_wing_age_std -1.325e-01 9.030e-02 -1.467  0.14711
## diff_back_age_std -5.825e-02 1.186e-01 -0.491  0.62510
## diff_pivot_age_std -2.707e-02 4.610e-02 -0.587  0.55917
## diff_gk_age_std   -4.658e-02 5.031e-02 -0.926  0.35798
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(2) family taken to be 0.03784458)
##
## Null deviance: 8.1676  on 100  degrees of freedom
## Residual deviance: 2.4923  on  65  degrees of freedom
## AIC: 883.07
##
## Number of Fisher Scoring iterations: 5

```

```

pred_model_NB_home <- predict(model_NB_home,X_test)

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## les prédictions venant d'un modèle de rang faible peuvent être trompeuses
pred_home <-exp(pred_model_NB_home)
# away
train_NB_away <- cbind(Y_train_away, X_train)
model_NB_away <- glm(away_score_final ~ .,data = train_NB_away,
                      family = negative.binomial(2))
summary(model_NB_away)

## 
## Call:
## glm(formula = away_score_final ~ ., family = negative.binomial(2),
##      data = train_NB_away)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.49195 -0.10247  0.00154  0.09189  0.38743
##
## Coefficients: (2 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.179e+00 3.206e+00  0.991  0.3252
## home_team_id -1.529e-03 4.245e-03 -0.360  0.7199
## away_team_id -1.494e-03 4.371e-03 -0.342  0.7337
## league_id        NA         NA       NA       NA
## home_travel_km      NA         NA       NA       NA
## away_travel_km -3.627e-06 6.341e-06 -0.572  0.5693
## home_nb_players -4.371e-03 2.122e-02 -0.206  0.8374
## home_clubs_ratio -3.229e-01 2.286e-01 -1.412  0.1626
## home_age_avg   1.359e-02 2.934e-01  0.046  0.9632
## home_age_std   2.673e-01 3.025e-01  0.883  0.3802
## home_height_avg -6.822e-02 1.508e-01 -0.452  0.6526
## home_height_std -2.828e-01 2.456e-01 -1.152  0.2536
## home_weight_avg -1.882e-02 3.147e-02 -0.598  0.5518
## home_weight_std 1.435e-02 1.991e-02  0.721  0.4736
## away_nb_players -2.056e-02 2.169e-02 -0.948  0.3467
## away_clubs_ratio 2.180e-01 2.258e-01  0.965  0.3379
## away_age_avg   -3.677e-02 3.006e-01 -0.122  0.9030
## away_age_std   -2.647e-01 2.992e-01 -0.885  0.3796
## away_height_avg 7.689e-02 1.474e-01  0.521  0.6038
## away_height_std 2.901e-01 2.469e-01  1.175  0.2442
## away_weight_avg 2.049e-02 3.341e-02  0.613  0.5419
## away_weight_std -1.903e-02 2.036e-02 -0.935  0.3533
## diff_wing_height_avg -5.098e-02 7.806e-02 -0.653  0.5160
## diff_back_height_avg -5.520e-04 8.108e-02 -0.007  0.9946
## diff_pivot_height_avg 6.693e-02 4.261e-02  1.571  0.1210
## diff_gk_height_avg  3.311e-02 1.793e-02  1.847  0.0693 .
## diff_wing_height_std 8.144e-02 9.522e-02  0.855  0.3956
## diff_back_height_std 1.189e-01 1.080e-01  1.101  0.2752
## diff_pivot_height_std 7.054e-02 6.244e-02  1.130  0.2628
## diff_gk_height_std  1.686e-02 2.573e-02  0.655  0.5146
## diff_wing_age_avg  2.915e-02 6.087e-02  0.479  0.6336

```

```

## diff_back_age_avg      -3.545e-02  9.955e-02  -0.356   0.7229
## diff_pivot_age_avg    -5.262e-03  5.339e-02  -0.099   0.9218
## diff_gk_age_avg       3.795e-04  6.359e-02   0.006   0.9953
## diff_wing_age_std    -5.084e-02  9.941e-02  -0.511   0.6108
## diff_back_age_std     -8.984e-02  1.306e-01  -0.688   0.4941
## diff_pivot_age_std   -6.660e-02  5.073e-02  -1.313   0.1939
## diff_gk_age_std       -8.328e-02  5.536e-02  -1.504   0.1374
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(2) family taken to be 0.0458475)
##
## Null deviance: 7.1771  on 100  degrees of freedom
## Residual deviance: 3.0427  on  65  degrees of freedom
## AIC: 881.98
##
## Number of Fisher Scoring iterations: 5
pred_model_NB_away <- predict(model_NB_away,X_test)

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## les prédictions venant d'un modèle de rang faible peuvent être trompeuses

pred_away <-exp(pred_model_NB_away)
# validation
true_home <- Y_test_home$home_score_final
true_away <- Y_test_away$away_score_final
pred <- c(pred_home[1],pred_away[1],pred_home[2],pred_away[2])
obs <- c(true_home[1],true_away[1],true_home[2],true_away[2])
postResample(pred = pred,
             obs = obs)

##      RMSE  Rsquared      MAE
## 4.2431028 0.3338491 3.6117010

# modèle gaussien
#home
model_Gaussien_home <- cv.glmnet(x = as.matrix(X_train), y = Y_train_home$home_score_final,
                                    type.measure = c("deviance"),
                                    family = c("gaussian"))
summary(model_Gaussien_home)

##           Length Class  Mode
## lambda      100   -none- numeric
## cvm         100   -none- numeric
## cvsd        100   -none- numeric
## cvup        100   -none- numeric
## cvlo        100   -none- numeric
## nzero       100   -none- numeric
## call         5    -none- call
## name         1    -none- character
## glmnet.fit  12   elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric

```

```

pred_model_Gaussien_home <- predict(model_Gaussien_home,as.matrix(X_test))
pred_home <- pred_model_Gaussien_home
# away
model_Gaussien_away <- cv.glmnet(x = as.matrix(X_train), y = Y_train_away$away_score_final,
                                    type.measure = c("deviance"),
                                    family = c("gaussian"))
summary(model_Gaussien_away)

##          Length Class  Mode
## lambda      100   -none- numeric
## cvm         100   -none- numeric
## cvsd        100   -none- numeric
## cvup        100   -none- numeric
## cvlo        100   -none- numeric
## nzero       100   -none- numeric
## call         5    -none- call
## name         1    -none- character
## glmnet.fit  12   elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric

pred_model_Gaussien_away <- predict(model_Gaussien_away,as.matrix(X_test))
pred_away <- pred_model_Gaussien_away
# validation
true_home <- Y_test_home$home_score_final
true_away <- Y_test_away$away_score_final
pred <- c(pred_home[1],pred_away[1],pred_home[2],pred_away[2])
obs <- c(true_home[1],true_away[1],true_home[2],true_away[2])
postResample(pred = pred,
             obs = obs)

##      RMSE  Rsquared      MAE
## 3.9964649 0.1138518 3.2813812

```