SY09 - Rapport de projet

Clément BRIZARD - Soukaina DIDI ALAOUI - Claire LE RAY

14 juin 2019

Introduction

Ce rapport est réalisé dans le cadre de l'UV SY09 - Analyse de données et *data mining*. Nous avons réalisé l'analyse d'un jeu de données grâce aux méthodes étudiées en cours, en utilisant la programmation en R.

Le jeu de données choisi s'intitule *Top 250 Football transfers from 2000 and 2018*, issu de la plate-forme Kaggle. Il liste les 250 transferts de joueurs les plus chers pour chaque saison entre 2000 et 2018 sous la forme d'un tableau individus-variables. Ce dernier contient 4700 lignes, et les variables disponibles sont :

- *Name* : nom du joueur ;
- *Position* : position du joueur ;
- *Age* : âge du joueur lors du transfert ;
- *Team_from* : équipe d'origine du joueur ;
- *Team_to* : nouvelle équipe du joueur ;
- *League_from* : ligue d'origine du joueur ;
- *League_to* : nouvelle ligue du joueur ;
- *Season* : saison ;
- *Market_value* : valeur estimée du joueur ;
- *Transfer_fee* : valeur réel du transfert.

Nous nous sommes intéressés à l'utilisation de ce *dataset* dans le cadre du *machine learning*. Nous sommes arrivés à la conclusion qu'il pouvait être utile aux clubs de football souhaitant vendre des joueurs. Aussi, nous nous sommes attachés à répondre aux questions suivantes :

Quelle plus-value puis-je espérer en vendant un joueur donné ? Quel profil de club pourrait être intéressé par mon joueur ?

Pour cela, nous avons réalisé un nettoyage du jeu de données, ainsi qu'une analyse exploratoire. Cette dernière nous a permis d'établir des hypothèses sur les données, ainsi que de construire des prédicteurs. Nous avons enfin utilisé différentes méthodes de *machine learning* pour répondre à nos questions. Ce rapport présente nos démarches et les résultats obtenus.

1 Analyse exploratoire

1.1 Analyse exploratoire

Le *dataset* contient les 250 plus gros transferts des saisons 2000-2001 à 2018-2019 (à moins de 10 transferts près par saison). Une première analyse de chaque variable permet de dégager plusieurs remarques, détaillées plus loin.

1.1.1 Position

Les attaquants sont les plus représentés [1], suivis des milieux de terrain et des défenseurs. Il y a dix fois plus d'attaquants que de gardiens parmi les transferts :

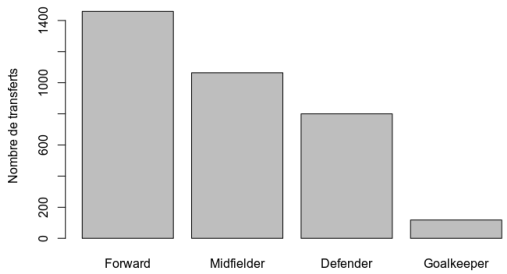

FIGURE 1 – Distribution des positions

1. Graphe réalisé après avoir agrégé les positions en quatre catégories, voir 1.2.5

1.1.2 League_from et League_to

Les ligues les plus représentées sont la **Premier League** (Angleterre) et la **Serie A** (Italie) à égalité, suivies de **LaLiga** (Espagne) et **Ligue 1** (France).

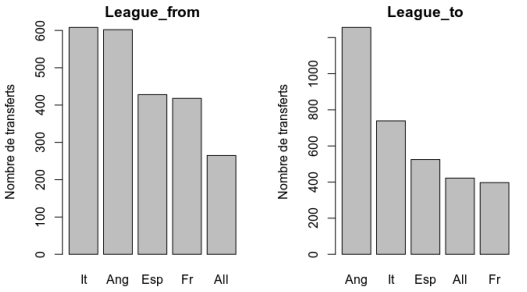

FIGURE 2 – Distribution des *League_from* et *League_to*

50% des transferts se sont faits depuis l'une de ces cinq ligues et 65% vers. On peut faire l'hypothèse que beaucoup de clubs peuvent vendre des joueurs chers (jeunes talents, etc.), mais peu de clubs disposent du budget pour les acheter, et ceux qui le peuvent sont concentrés dans les ligues les plus importantes.

1.1.3 Variables quantitatives

Le *dataset* comprend 3 variables quantitatives : *Age*, *Market_value* et *Transfer_fee*. Une analyse statistique permet d'obtenir les données suivantes :

Valeurs	Age	Market_value	Transfer_fee
Min	0.00	50000	825000
Q1	22	3500000	4000000
Médiane	24 ans	6M	6.5M
Mean	24.34	8622469	9447586
Q3	27	10000000	10820000
Max	35	120M	222M
NA's	-	1260	-

Il est possible de résumer ces valeurs par le graphique suivant :

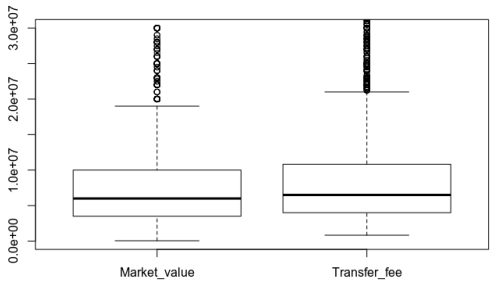

FIGURE 3 – Boxplots comparés de *Market_value* et *Transfer_fee*

Ces graphiques nous permettent d'observer que *Transfer_fee* semble toujours plus élevée que *Market_value*. Il y a également plus de variance au dessus de la médiane dans les deux cas. On remarque également un grand nombre de valeurs aberrantes pour les deux variables : la médiane est donc un indicateur plus fiable que la moyenne des valeurs.

On s'intéresse ensuite à la distribution de ces variables. La réalisation d'histogramme nous permet de faire l'hypothèse que *Age* suit une loi normale. Cela n'est pas le cas pour les deux autres variables quantitatives :

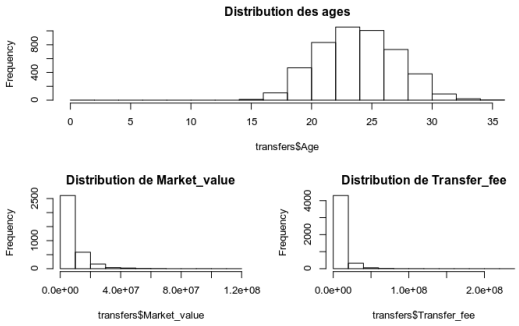

FIGURE 4 – Distribution des variables quantitatives

Constatant l'écart entre *Market_value* et *Transfer_fee*, nous décidons d'orienter notre travail de *machine learning* sur la prédiction de cet écart.

1.2 Nettoyage du *dataset*

Le jeu de données issu de *Kaggle* contient 4700 transferts. Il doit contenir les 250 transferts les plus chers pour chaque saison entre les saisons 2000-2001 et 2018-2019. Toutefois, on s'aperçoit rapidement qu'il n'y a pas exactement 250 transferts pour chaque saison. Néanmoins, il en manque au plus 8 pour une saison donnée : les résultats ultérieurs ne devraient donc pas être faussés.

On dispose pour chaque transfert des 10 prédicteurs présentés en introduction. Parmi eux, 7 sont des facteurs, et 3 des variables numériques continues :

```
        Name     Position          Age    Team_from
    "factor"     "factor"    "integer"     "factor"
Market_value Transfer_fee
   "integer"    "integer"
League_from      Team_to    League_to       Season
    "factor"     "factor"     "factor"     "factor"
```

FIGURE 5 – Classes des variables du jeu de données

Pour permettre une analyse exploratoire efficace et faciliter les traitements ultérieurs, on souhaite :

- Déterminer le traitement à apporter aux valeurs manquantes (`NA`) ou incohérentes ;
- Enrichir le jeu de données à partir de variables composites adaptées à nos besoins ;
- Sélectionner les variables explicatives pertinentes ;
- Limiter le nombre de modalités des variables qualitatives.

1.2.1 Valeurs manquantes

Une étude rapide du jeu de données permet de déterminer que seule la colonne *Market_value* contient une grande quantité de `NA`, à raison de 1260 sur 4700, soit 27%. La réalisation d'une table de contingence permet de déterminer que ces valeurs manquantes sont concentrées à plus de 90 % sur les 5 premières saisons étudiées :

```
2000-2001 2001-2002 2002-2003 2003-2004 2004-2005 2005-2006 2006-2007
      248       250       244       242       189        28        20
2007-2008 2008-2009 2009-2010 2010-2011 2011-2012 2012-2013 2013-2014
       13         7         2         4         1         2         2
2014-2015 2015-2016 2016-2017 2017-2018 2018-2019
        1         0         1         3         3
```

FIGURE 6 – Table de contingence

Dans la mesure où nous avons choisi de nous intéresser à la prédiction de l'écart entre *Market_value* et *Transfer_fee*, nous avons besoin de la valeur de la colonne *Market_value*. Nous décidons donc de supprimer les transferts où cette valeur est nulle. Le *dataset* contient encore 3440 transferts, lesquels sont, qui plus est, les plus récents.

1.2.2 Valeurs incohérentes

Seules les ligues présentaient des valeurs ambiguës, en voici quelques exemples, sur lesquels nous avons fait une vérification en nous documentant :

- "Série A" correspond à la première division brésilienne et "Serie A" à l'italienne ;
- "Bundesliga" correpond à la première division autrichienne et "1. Bundesliga" à l'allemande.

1.2.3 Enrichissement du jeu de données

On souhaite travailler sur la plus-value réalisée lors de la vente d'un joueur : il s'agit de la différence entre la valeur du transfert réelle (variable `Transfer_fee`) et la valeur du joueur estimée sur le marché (variable `Market_value`). On construit ainsi facilement cette colonne en R, et on l'ajoute au jeu de données sous le nom `plus_value`.

De plus, devant le grand nombre de modalités des variables contenant des ligues (`League_from` et `League_to`) ou des équipes (`Team_from` et `Team_to`), on souhaiterait établir des profils de ligues et de clubs ayant du sens et permettant de regrouper les données. Le traitement réalisé est détaillé en section 2.

1.2.4 Sélection des variables explicatives pertinentes

Dans un jeu de données contenant beaucoup de variables, il est important de sélectionner

celles ayant le plus de sens au regard des objectifs d'analyse et de traitement fixés.

On supprime la variable `Name`, contenant les noms des joueurs et qui comporte trop de modalités en apportant peu d'information. En effet, un joueur pouvant évoluer (âge, position, équipe, valeur sur le marché), conserver cette variable pourrait entraîner le rapprochement erroné de points du *dataset*.

On supprimera également les colonnes `Team_from`, `Team_to`, `League_from` et `League_to` après leur factorisation détaillée section 2.

Enfin, on supprime la colonne *Transfer_fee*, car nous souhaitons uniquement prédire la plus-value réalisée.

1.2.5 Réduction du nombre de modalités

Du fait de la grande quantité de variables qualitatives, on souhaite réduire leur nombre de modalités.

On s'intéresse d'abord à la variable qualitative *Positions*. Celle-ci comprend initialement 17 niveaux. On souhaite les regrouper en 4 niveaux :

- Milieu (*Midfielder*) ;
- Défenseur (*Defender*) ;
- Avant(*Forward*) ;
- Gardien (*Goalkeeper*).

A partir de connaissances expertes issues de l'encyclopédie collaborative Wikipédia, on opère les regroupements suivants :

- **Midfielder :** *Attacking Midfield*, *Central Midfield*, *Defensive Midfield*, *Left Midfield*, *Midfielder* et *Right Midfield* ;
- **Defender :** *Centre-Back*, *Left-Back*, *Right-Back* et *Sweeper* ;
- **Forward :** *Centre-Forward*, *Forward*, *Second Striker*, *Right Winger*, et *Left Winger* ;
- **Goalkeeper**.

On utilise pour cela des expressions régulières sur les niveaux initiaux.

Les modalités de ligues et d'équipes ont été regroupés par classification automatique.

2 Clustering des ligues et équipes

2.1 Des variables intéressantes à condition d'être agrégées

On veut qu'un propriétaire de club puisse avoir une prédiction de la plus-value qu'il peut obtenir d'un joueur. Au-delà de la position de son joueur, et de son âge, le propriétaire sait que tous les clubs ne réalisent pas les mêmes types de transferts. Informée par l'analyse exploratoire, notre intuition est que certains clubs très riches peuvent se permettre d'acheter plus qu'ils ne vendent, tandis que d'autres clubs plus modestes cherchent à vendre des joueurs qu'ils ont souvent formés, à un prix élevé pour réaliser une forte plus-value. Le propriétaire doit ainsi pouvoir se situer parmi les types de clubs. Il en est de même pour les ligues.

Autrement dit, le club et la ligue d'origine, ainsi que le club et la ligue d'arrivée vont avoir une importance dans notre apprentissage, à condition que nous les regroupions. Actuellement, nous disposons en effet de pas moins de 467 valeurs de clubs uniques et 81 valeurs de ligues.

2.2 Méthode : K-means

On développe l'explication pour les équipes, la procédure sera la même pour les ligues.

Nous souhaitions que la classification des clubs ait du sens, en termes de comportements d'achats et de ventes.

On commence par agréger les clubs en trois tableaux[2] :

- les clubs ayant à la fois acheté et vendu des joueurs : les "vendeurs-acheteurs". 224 clubs. On calcule pour chaque équipe la différence entre la somme de ses ventes et de ses achats. En tête des clubs ayant réalisé le plus de pertes, on retrouve Manchester City (Angleterre) et le Real Madrid (Espagne), deux équipes connues pour battre chaque année le record du plus gros transfert jamais réalisé. Au contraire, on retrouve en tête des clubs réalisant le plus de plus-value

2. Cette démarche nous a été inspirée par un *kernel Kaggle* développé sur notre dataset.

des clubs portugais comme le Benfica Lisbonne et le FC Porto, connus pour miser leur stratégie sur la formation de joueurs qu'ils revendront ensuite à un prix élevé. [3]
— les clubs ayant uniquement vendu des joueurs : les "vendeurs". On calcule pour chaque club la somme de ses ventes. 205 clubs.
— les clubs ayant uniquement acheté des joueurs : les "acheteurs". On calcule pour chaque club la somme de ses achats. 38 clubs.

On souhaite alors déterminer le nombre de classes optimal pour chacun de ces groupes. On applique pour ce faire la méthode vue dans le TP5 de SY09 :

1. effectuer 100 classifications en prenant K = 2 classes, puis à nouveau 100 classifications avec K = 3 classes, K = 4 classes, *etc.* jusqu'à K = 10 classes ;

2. pour chaque valeur de K, on calcule l'inertie intra-classe minimale sur les 100 répétitions ;

3. on représente graphiquement l'inertie intra-classe minimale en fonction du nombre de classes, et on choisit le nombre de classes optimal (méthode du coude).

Dans notre cas, l'algorithme K-means opérait sur une seule variable : la plus-value pour les clubs vendeurs-acheteurs, la somme des achats pour les acheteurs, et la somme des ventes pour les vendeurs.

À titre d'exemple, voici le graphe obtenu pour l'inertie intra-classe minimale selon le nombre K de classes pour les clubs vendeurs-acheteurs :

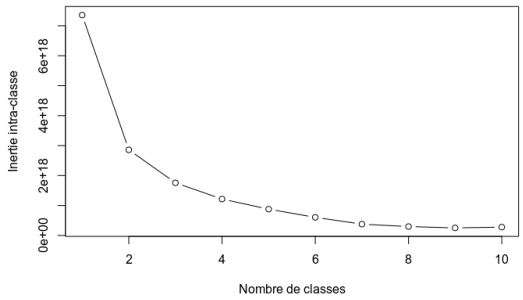

FIGURE 7 – Inertie intra-classe minimale en fonction du nombre de classes pour les clubs vendeurs-acheteurs

En choisissant K = 4 classes, nous obtenions le clustering suivant pour les clubs vendeurs-acheteurs :

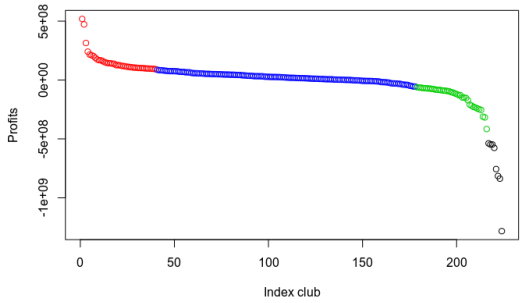

FIGURE 8 – Clustering des clubs vendeurs-acheteurs pour K = 4 classes

En procédant de même pour les deux autres types de clubs, nous avons choisi 3 niveaux de clubs vendeurs, et 2 de clubs acheteurs, soit 9 modalités au lieu de 467. En fusionnant nos clubs étiquetés avec le tableau des transferts, nous disposions maintenant, dans chaque transfert, d'un type pour le club d'origine et le club d'arrivée.

Nous avons procédé de même pour les ligues, réduisant le nombre de modalités de 81 à 7 (3 niveaux de ligues "vendeuses-acheteuses", 3 de ligues "vendeuses" et 1 de ligue "acheteuses", car uniquement 5 ligues concernées).

Nous supprimons alors les colonnes des équipes et des ligues. À l'issue de

3. Ces assertions sont celles d'un membre de notre groupe, qui n'est peut-être pas un "expert" du sujet, mais s'appuie néanmoins sur 15 ans de pratique du football dont deux ans en centre de formation.

notre clustering, nos algorithmes d'apprentissage vont donc désormais pouvoir bénéficier de 4 nouvelles variables qualitatives : *Type_ Team_ from* et *Type_ Team_ to* (9 modalités) et *Type_ League_ from* et *Type_ League_ to* (7 modalités).

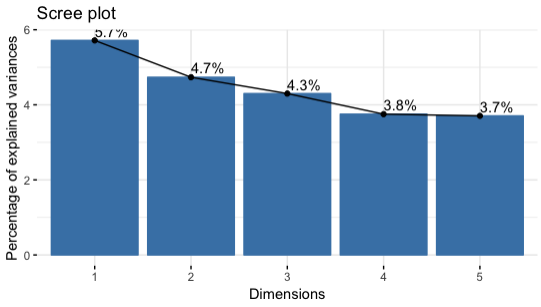

FIGURE 9 – Proportions de variances expliquées par les différents axes

3 Analyse factorielle

Après le nettoyage de notre jeu de données, ce dernier est toujours sous forme d'un ensemble d'individus (3440 observations) décrits par des données multiples (variables qualitatives et quantitatives factorisées). Il reste donc complexe : par conséquent, nous avons décidé d'appliquer des méthodes d'analyse factorielle afin de mieux le résumer et le visualiser.

Les méthodes qui nous permettent cela sont :

— MFA (*Multiple Factor Analysis*) ;
— AFDM (Analyse Factorielle des Données Mixtes).

La méthode MFA impose d'avoir des groupes de notre ensemble de variables, tandis que la méthode AFDM permet de faire une analyse des variables qualitatives et quantitatives mélangées sans contraintes : de ce fait, nous jugeons que l'AFDM est plus adaptée à notre cas.

Afin d'appliquer cette méthode, nous avons eu recours aux librairies `FactoMineR` et `factoextra`. Nous utilisons d'abord la fonction R `FAMD`, qui permet d'appliquer la méthode AFDM à notre jeu de données.

Nous utilisons ensuite la fonction `fviz_screeplot` pour visualiser les proportions de variances expliquées par les différents axes :

Nous remarquons que l'inertie est expliquée quasi-équitablement par les différentes dimensions de la FAMD. Par conséquent, nous allons chercher à obtenir la qualité de représentation et les contributions de chaque variable. Dans un premier temps, nous allons montrer, dans la figure suivante, la corrélation entre les variables et les dimensions principales ainsi que la contribution des variables aux 2 premiers axes :

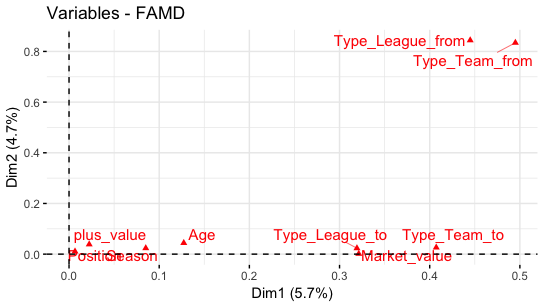

FIGURE 10 – Corrélation entre variables et axes et contributions aux 2 premiers

Nous pouvons donc remarquer que les variables qualitatives relatives aux ligues et aux équipes d'origine (*from*) des joueurs contribuent en majorité aux deux premières dimensions. Tandis que celles de destination des joueurs (*to*) en plus de la variable quantitative *Market_ value* sont les plus corrélées avec la première dimension et y contribuent suffisamment mais ne contribuent quasiment pas à la deuxième dimension. Les variables restantes sont un peu corrélées avec la première dimension mais ne contribuent quasiment à aucune.

Pour une meilleure visualisation et appré-

hension de la contribution des autres variables, nous avons tracé la contribution des variables par rapport à chaque axe. Ces figures sont placées en **Annexe 1**. Ces figures nous ont permis de voir comment certaines variables contribuent beaucoup à des dimensions précises et très peu voire quasiment pas à d'autres : pour s'en convaincre, nous avons vérifié si la moyenne attendue (valeur atteinte si l'ensemble des contributions était uniforme et qui est le trait en pointillé rouge sur chaque figure) est atteinte ou pas par la contribution du reste des variables. Or ce n'était pas le cas.

4 Classification automatique hiérarchique agglomérative

Après la réalisation de l'analyse factorielle, nous avons décidé d'exploiter les résultats de cette dernière en appliquant une classification hiérarchique ascendante avec l'approche agglomérative. Cette méthode consiste à placer initialement chaque point du *dataset* dans un *cluster*, puis à regrouper les points plus proches (on parle alors d'approche *bottom-up*). Afin de réaliser cela, nous avons eu recours à la fonction R `HCPC` (Hierarchical Clustering on Principle Components) qui exploite automatiquement les données de l'analyse factorielle passée en paramètre et nous renvoie des résultats contenant nos clusters avec leurs représentations dont la carte des *clusters* en 3D ci-dessous :

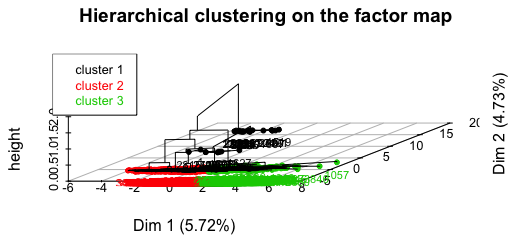

FIGURE 11 – *Clusters* 3D réalisés grâce à la fonction `HPCP`

Cette méthode nous propose donc de regrouper nos individus dans 3 *clusters* différents. Plus tard, dans notre travail, nous avons décidé de ne pas utiliser le résultat de ce *clustering* puisque nos données nous semblent être assez bien factorisées après les deux premières classifications automatiques sur les équipes et les ligues.

5 Apprentissage supervisé

On souhaite réaliser une régression sur le paramètre `plus_value` construit à partir des variables `Transfer_fee` et `Market_value`. Pour cela, on dispose du reste des prédicteurs du jeu de données. On décide donc d'appliquer différentes méthodes d'apprentissage supervisé issues du cours de SY09.

Pour entraîner un classifieur en utilisant ces algorithmes et mesurer l'erreur commise, il est nécessaire de diviser l'ensemble des données `transfers` en deux sous-ensembles :

- L'**ensemble d'apprentissage**, contenant les données utilisées pour établir un modèle de prédiction ;
- L'**ensemble de test**, contenant les données utilisées pour estimer le risque du modèle.

Pour maximiser la quantité de données sur laquelle est entraîné le classifieur et pour déterminer plus efficacement son taux d'erreur de prédiction, on décide d'utiliser la **validation croisée** : cette méthode consiste à partitionner le *dataset* initial $\mathcal{D}$ en N_V blocs $\mathcal{D}_1, ..., \mathcal{D}_{N_V}$. On entraîne alors le classifieur sur $\mathcal{D} \backslash \mathcal{D}_k$, pour chaque $k \in \{1, ..., N_V\}$. On utilise le sous-ensemble restant $\mathcal{D}_k$ comme ensemble de test.

On peut alors entraîner le classifieur sur l'ensemble des données, tout en obtenant k estimateurs de risques non biaisés dont on peut alors faire la moyenne pour évaluer l'erreur commise sur $\mathcal{D}$.

Pour ce projet, nous avons choisi $k = 10$.

5.1 Adaptation du jeu de données

À l'exception de la régression linéaire, la majorité des modèles d'apprentissage supervisé étudiés en SY09 permettent de résoudre des problèmes de classification. Aussi, pour pouvoir les appliquer au jeu de données, il était néces-

saire de découper les valeurs de la variable qualitative `plus_value` en facteurs.

Se pose donc la question du découpage à effectuer. On s'intéresse d'abord à la règle de Sturges, une règle empirique permettant d'évaluer le nombre K de facteurs à créer :

$$K = 1 + \frac{10}{3} \log_{10} n$$

Toutefois, du fait de l'écart-type important entre les données, appliquer cette règle crée des classes de faible densité, comme on le voit dans le graphique ci-dessous :

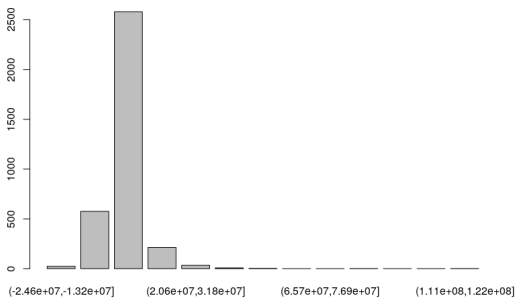

FIGURE 12 – Densité des classes avec un découpage régulier selon la règle de Sturges

De plus, un découpage régulier n'a pas forcément de sens du point de vue de l'utilisation des données. On décide donc de se concentrer sur cet aspect.

En effet, on peut supposer que l'important pour un club souhaitant vendre un joueur est de ne pas faire de "mauvaise affaire", c'est-à-dire de réaliser une plus-value négative. Aussi, on divise la variable `plus_value` en deux classes, séparant les plus-values **positives** et **négatives**. On obtient le graphique suivant, plus équilibré :

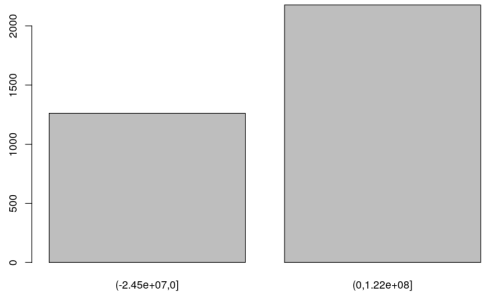

FIGURE 13 – Densité avec un découpage binaire

Finalement, notons que les deux variables quantitatives ont été normalisées avant d'appliquer les différents modèles de l'apprentissage supervisé.

5.2 Régression linéaire

On est dans le cas d'un problème de régression ; aussi, on s'intéresse d'abord à un modèle de régression linéaire multiple.

La régression linéaire multiple a pour but l'étude de la relation entre une variable à expliquer quantitative Y (ici `plus_value`) et p variables explicatives $x_1, ..., x_p$ (ici le reste du *dataset*).

Toutefois, du fait du grand nombre de variables qualitatives dans le jeu de données, cette méthode est peu adaptée : en effet, ces dernières ne sont pas intégrables à une équation linéaire.

On décide de s'intéresser à la régression linéaire de `plus_value` en fonction des variables qualitatives `age` et `Market_value`. Pour cela, on décide d'afficher un graphe de cette modélisation.

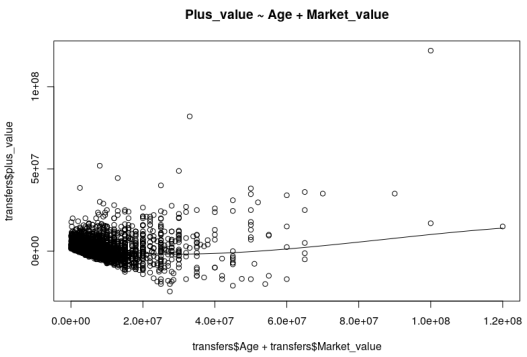

FIGURE 14 – Courbe lissée issue du nuage de points formé par les données

Le diagramme de dispersion associé à la ligne de lissage ci-dessus suggère une relation à croissance très faible que nous pourrions moyennement qualifier de linéaire entre les variables "Age + Maket_value" et "Plus_value". Ceci n'est pas un bon signe puisque l'une des hypothèses de la régression linéaire est que la relation entre la réponse et les variables prédictives est linéaire et additive. Nous avons aussi vérifié que la variable réponse (Plus_value) suivait bien une loi normale avec le Shapiro-Wilk sur R. De plus, nous avons vérifié la corrélation entre les prédicteurs ("Age + Maket_value") et la réponse "Plus_value" avec la fonction R `cor` qui nous a renvoyé des valeurs appartenant à l'intervalle [-0,2,0,2]. Cela indique une faible corrélation entre les deux, donc il est possible que les prédicteurs n'expliquent pas beaucoup la variation de la réponse à la variable Plus_value. Par conséquent, il est possible que ces prédicteurs ne soient pas les meilleures variables explicatives a utiliser. De ces résultats, nous pouvons déduire que les hypothèses de la régression linéaire ne sont pas vraiment respectées.

En dépit des résultats trouvés ci-dessus, nous avons appliqué ce modèle à notre jeu de données, nous avons utilisé, avec la *cross-validation*, la fonction R `lm` (Fitting Linear Models) avec une deuxième fonction `step` nous permettant de faire de façon récursive une sélection de variables en s'appuyant sur le critère AIC (minimiser la fonction de vraisemblance) pour avoir les variables significatives et donc pour éviter l'*overfitting*. Un *summmary* du résultat nous a permis de vérifier que les variables sont significatives (pvalue < 0.05) ainsi que la régression. Or l'erreur de ce modèle était de l'ordre de : 2.866465e+13 ce qui est très grand ! Nous avions soupçonné cela quand on a vérifié les hypothèses de ce modèle.

5.3 Régression logistique

Aussi, on décide d'entraîner un nouveau classifieur, en utilisant la régression logistique. Cette méthode consiste à modéliser les probabilités *a posteriori* des différentes classes. Elle fonctionne de manière similaire à la régression linéaire, mais ne nécessite pas de relation linéaire entre les variables. De plus, il est possible de conserver les variables qualitatives : ce modèle semble donc plus adapté à notre jeu de données. Tout de même, avant de l'appliquer nous vérifions le biais de la classe à prédire. Avec un *summary* sur celle-ci nous obtenons : 0 : 1263 et 1 : 2177 , donc nous pouvons dire que nous avons un biais de classe mais la différence entre les deux grandeurs est de 914, ce qui ne représente que 26.57% du nombre total d'individus. Par conséquent, nous avons décidé de poursuivre et appliquer ce modèle.

5.4 Classifieur bayésien naïf

On s'intéresse ensuite au classifieur bayésien naïf. Celui-ci fait appel à la théorie bayésienne de la décision, qui permet de créer une règle de décision en minimisant l'erreur commise : celle-ci consiste à choisir la classe de **plus grande probabilité *a posteriori***.

Toutefois, le calcul de cette quantité suppose de connaître la distribution des probabilités dans la population totale. Pour contourner cette difficulté, on utilise l'**hypothèse naïve**, c'est-à-dire que l'on suppose que les variables explicatives sont indépendantes entre elles, ce qui permet de réaliser une approximation de leurs probabilités conditionnelles.

Dans le cas de notre jeu de données, cette approximation paraît difficilement acceptable. En effet, tant l'intuition que les calculs de corrélation effectuées tendent à montrer que les variables explicatives du *dataset* ne sont pas indépendantes.

5.5 Arbres de décision

On s'intéresse ensuite aux différents algorithmes permettant la modélisation d'arbres de décision. Il s'agit de la seule méthode étudiée en SY09 permettant de gérer des variables corrélées : aussi, on s'attache particulièrement à l'étude de ceux-ci.

5.5.1 Arbre binaire : algorithme CART

On cherche d'abord à réaliser un arbre binaire permettant de prédire la plus-value. Ce procédé consiste à établir une série de tests sur les prédicteurs, chaque test permettant de construire un nœud ayant 0 ou 2 fils. Le processus se termine lorsqu'on atteint l'une des feuilles de l'arbre : celle-ci correspond à une valeur de la variable à expliquer (ici `plus_value`).

Pour déterminer les divisions à effectuer, on utilise l'algorithme CART, implémenté grâce à la libraire `rpart` de R. Celle-ci implémente une stratégie de construction d'un arbre binaire, en utilisant l'indice de Gini comme critère d'impureté.

$$G(\mathbf{p}) = \sum_{k=1}^{g} p_k(1 - p_k)$$

Toutefois, cette méthode est peu adaptée aux variables qualitatives, notamment dans le cas d'un grand nombre de modalités. En effet, la binarité de l'arbre entraîne une difficulté à séparer celles-ci, ce qui a un effet négatif sur la complexité de l'arbre obtenu. De plus, il est important de gérer au mieux l'élagage de l'arbre, pour obtenir un modèle performant tout en évitant le sur-apprentissage (une feuille par observation).

On choisit donc de s'intéresser à d'autres algorithmes permettant la construction d'arbres de décision.

5.5.2 Algorithme C4.5

On a vu précédemment que du fait de la grande quantité de modalités dans les variables du jeu de données, un arbre binaire est peu adapté. Aussi, on s'intéresse à un autre algorithme permettant d'aboutir à un arbre de décision : l'algorithme C4.5, défini par Ross Quinlan.

Contrairement aux arbres construits grâce à l'algorithme CART, les nœuds d'un arbre construit grâce à l'algorithme C4.5 peuvent posséder un nombre quelconque de fils.

Ici, le critère d'impureté utilisé est l'entropie.

$$E(\mathbf{p}) = -\sum_{k=1}^{g} p_k \ln p_k$$

L'entropie agit de manière similaire à l'indice de Gini : toutefois, l'utilisation de logarithmes implique que les probabilités les plus faibles sont mieux prises en compte en utilisant l'entropie. Dans le cas de multiples modalités de faible densité, l'entropie peut donc représenter une meilleure solution.

Lors de la première exécution de l'algorithme (implémenté grâce à la librairie `RWeka`), on a rapidement observé que l'arbre produit ne contenait qu'une feuille : l'arbre revenait donc à un vote majoritaire. On décide donc de régler manuellement les paramètres de *post-pruning* : on règle ainsi le niveau de confiance C dans le jeu de données. Celui-ci influe sur le calcul de l'erreur estimée de chaque nœud, et donc sur la décision d'élaguer ceux-ci.

De ce fait, on comprend rapidement qu'un niveau de confiance faible donne un arbre trop simple, ne permettant pas une bonne précision, tandis qu'un niveau de confiance élevé entraîne une baisse des performances liées au sur-apprentissage.

Ici, la valeur $C = 0.28$ semble permettre les meilleures performances.

5.5.3 Algorithme C5.0

On teste enfin une évolution de l'algorithme C4.5, appelé C5.0. Celle-ci produit des arbres plus petits et permet une meilleure gestion de la mémoire dans le cas d'un jeu de données de grande taille.

Comme pour l'algorithme précédent, on règle également le niveau de confiance C : ici, la valeur optimale semble être $C = 0.32$.

5.5.4 *Random Forest*

Enfin, on s'intéresse à une évolution des arbres de décision : le *Random Forest Classifier*, c'est-à-dire une forêt d'arbres décisionnels.

Cette méthode consiste à réaliser un **vote majoritaire** sur de multiples arbres de décision, chacun ayant été appris sur un ensemble de données légèrement différent.

On mêle ainsi les concepts de ***bagging*** (combinaison de plusieurs classifieurs) et de **sous-ensemble aléatoire** (sélection de plusieurs ensembles d'apprentissage différents).

6 Résultats

6.1 Analyse des performances

On a regroupé l'ensemble des mesures d'erreur des algorithmes utilisés :

Algorithme	Erreur
Régression linéaire	2.866465e+13
Régression logistique	0.2743716
Classifieur bayésien naïf	0.2864167
Arbre binaire	0.2428844
C4.5	0.2670414
C5.0	0.2596045
Random Forest	0.2484483

FIGURE 15 – Tableau récapitulatif des erreurs commises

L'ensemble des résultats ont été obtenus grâce à la validation croisée, qui a amélioré nettement les performances.

On constate que la régression linéaire présente de très mauvaises performances : cela est cohérent, pour les raisons expliquées plus haut.

Pour les autres algorithmes, les différences sont moins flagrantes. On observe tout de même que le meilleur résultat est obtenu grâce à l'algorithme *Random Forest* : cela est cohérent, car cette méthode repose sur le *bagging* et le ré-échantillonage aléatoire, qui améliore ses performances par rapport à un arbre de décision simple. Toutefois, l'arbre binaire fonctionne également bien : on peut supposer que l'élagage y était optimal.

On notera également l'importance du choix des variables explicatives : celui-ci explique notamment l'échec de la régression linéaire. De plus, au fur et à mesure de l'avancée de notre travail de nettoyage du *dataset* et de classification automatique, nous avons observé une nette amélioration des performances.

6.2 Pistes d'amélioration

Bien que nous ayons utilisé un grand nombre des méthodes étudiées en cours de SY09 dans ce projet, d'autres améliorations sont possibles pour approfondir nos résultats et améliorer les performances.

Tout d'abord, dans le contexte d'une utilisation professionnelle d'un travail de *machine learning* sur ces données, l'établissement d'un cahier des charges complet eut été indispensable. En effet, il est important d'établir de manière précise l'utilisation souhaitée des données récoltées. Si nous avons établi en début de projet nos objectifs, un échange avec les utilisateurs finaux aurait enrichi nos travaux.

De la même manière, pour l'apprentissage supervisé, le découpage des classes à prédire lors de l'utilisation d'algorithmes de classification est largement améliorable. Nous avons choisi un modèle simple permettant de diviser le *dataset* en "bonnes" et "mauvaises" affaires. Toutefois, on peut supposer que la réalité est autrement plus nuancée et aurait nécessité de créer plus de classes ayant du sens d'un point de vue métier. Pour cela, deux solutions sont envisageables : l'intégration de connaissances expertes ou un *clustering* similaire à celui effectué sur les équipes et les ligues.

Enfin, de nombreux autres algorithmes pourraient permettre de réaliser une prédiction sur `plus_value` : nous nous sommes concentrés sur les algorithmes étudiés en cours, mais il existe d'autres possibilités, reposant notamment sur le ***boosting*** ou le ***bootstrap***.

Conclusion

En définitive, ce projet nous a permis d'appliquer les concepts et méthodes étudiés en cours de SY09 à un jeu de données réel. Nous avons ainsi eu l'opportunité de déterminer nos propres objectifs pour extraire des informations utiles de notre *dataset* : ce fut une motivation supplémentaire et une démonstration de la puissance de l'analyse de données.

Ce projet nous a également permis de nous confronter à la réalité d'un jeu de données. Nous avons en effet mesuré l'importance du nettoyage des données ainsi que du choix des variables explicatives pertinentes pour construire un modèle adapté aux objectifs fixés. Ces derniers sont en effet particulièrement importants pour orienter l'ensemble des travaux effectués sur les données : il est donc nécessaire de les déterminer avec précision.

Enfin, nous avons amélioré notre maîtrise de la programmation en R et découvert quelques-unes de ses nombreuses librairies. Si celles-ci nous ont permis d'implémenter facilement différents algorithmes d'analyse, de classification et de régression, nous avons réalisé l'importance de comprendre ces derniers pour pouvoir les appliquer au mieux. L'enseignement dispensé en SY09 nous a ainsi apporté la compréhension de ces concepts et nous a permis de les utiliser de manière éclairée.

Références

[1] Anastasia Reusova (2018). Hierarchical Clustering on Categorical Data in R. *Disponible en ligne à : this URL.*

[2] kassambara (2017). MFA - Multiple Factor Analysis in R : Essentials. *Disponible en ligne à cette URL.*

[3] kassambara (2017). AFDM - Analyse Factorielle des Données Mixtes avec R : L'Essentiel *Disponible en ligne à cette URL.*

[4] J.PAGÈS (2002). Analyse factorielle multiple appliquée aux variables qualitatives et aux données mixtes *Disponible en ligne à cette URL.*

[5] G. Govaert, T. Denoeux, B. Quost. SY09 Printemps 2019. TP 5 - Classification automatique.

[6] slehkyi. *football-transfers Disponible en ligne à cette URL.*

[7] Jason Brownlee *football-transfers Disponible en ligne à cette URL.*

[8] Wikipédia *Association football positions Disponible en ligne à cette URL.*

[9] Ricco Rakotomalala *Clustering of categorical variables Disponible en ligne à cette URL.*

Annexe 1 : contribution des variables aux axes