COMPTE RENDU DU TP3 :

# *Prédiction de la demande de vélos partagés*

Amélie Segard, Léa Bresson

---

## Le problème étudié

On cherche à **prédire le nombre de vélos partagés**, type « *Vélib* », loués par heure à Washington. Le jeu de données disponible contient le nombre de vélos loués dans la ville, par des utilisateurs abonnés et non abonnés, chaque heure, du 1er au 19ème jour du mois pour les années 2011 et 2012[1]. Des données sur les conditions météorologiques et des variables temporelles sont également disponibles chaque heure.

Nous allons utiliser des méthodes relatives aux problèmes de régression afin de modéliser la demande de vélo par des utilisateurs abonnés et non abonnés.

Le présent document présente successivement

1) Les données disponibles
2) L'exploration des données
3) La modélisation utilisée
4) Le critère de performance utilisé
5) L'ajustement des modèles
6) Les résultats

Nous présentons également en annexe deux méthodes d'ensemble employées lors de la modélisation : l'algorithme des Forêts aléatoire et l'algorithme Gradient Boosting.

---

[1] **Note :** Le jeu de données provient de la compétition Kaggle « *Bike Sharing Demand* ». Deux jeux de données sont disponibles : la base d'apprentissage (du 1er au 19ème jour du mois) et la base de test (du 20ème jour jusqu'à la fin du mois). La base de test ne contient pas la variable d'intérêt (i.e. la demande de vélo), ainsi nous ne pouvons pas l'utiliser dans notre rapport. Par conséquent, nos données concernent seulement la base d'apprentissage présente sur Kaggle. Afin d'implémenter une validation croisée, nous suivons une méthodologie similaire à celle de Kaggle qui consiste à diviser la base de données en base d'apprentissage et en base de test en tenant compte de la temporalité (début du mois versus fin du mois).

## 1) Les données disponibles

Le jeu de données est un ensemble de relevés de vélos partagés (type « *Vélib* »). Voici une brève description des données :

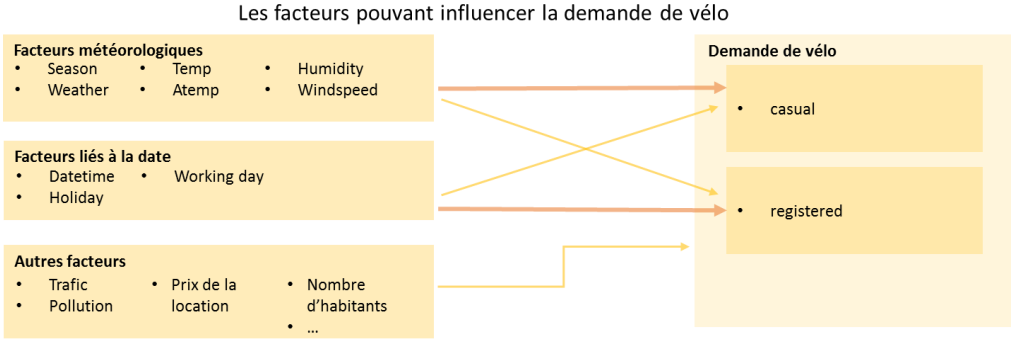
**datetime** – date et heure du relevé
**season** – 1 = printemps, 2 = été, 3 = automne, 4 = hiver
**holiday** – indique si le jour est un jour de vacances scolaires
**workingday** – indique si le jour est travaillé (ni week-end, ni vacances)
**weather** – 1 = dégagé à nuageux, 2 = brouillard, 3 = légère pluie ou neige, 4 = fortes averses ou neiges
**temp** – température en degrés Celsius
**atemp** – température ressentie en degrés Celsius
**humidity** – taux d'humidité
**windspeed** – vitesse du vent
**casual** - nombre de locations d'usagers non abonnés
**registered** – nombre de locations d'usagers abonnés
**count** – nombre total de locations de vélos

Ainsi, nous disposons de deux variables d'intérêt, à savoir les variables *Casual* et *Registered ;* celles-ci représentent la demande de vélos selon le statut de l'utilisateur (abonné au service de location de vélo ou non abonné).

Nous avons divisé la variable *datetime* en 4 variables: year, month, day, hour. Par ailleurs, le jeu de données présente 58 lignes manquantes sur 10 887, ce que l'on considère comme négligeable. (Une méthode d'imputation des données ne semble pas pertinente ici car les données manquantes restent marginales et de surcroît nous avons seulement des informations sur datetime pour ces lignes.)

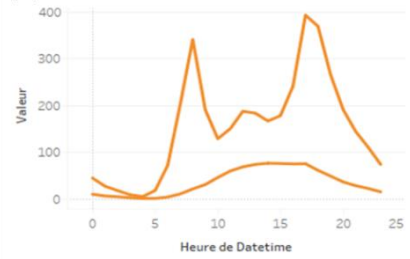
## 2) Exploration des données

Intuitivement, nous pouvons penser que les facteurs clés déterminant la demande de vélos pour un usager abonné et un usager non abonné ne sont pas les mêmes.
Le schéma ci-dessous formalise nos intuitions et recense ces facteurs et l'importance de leur effet sur la demande.

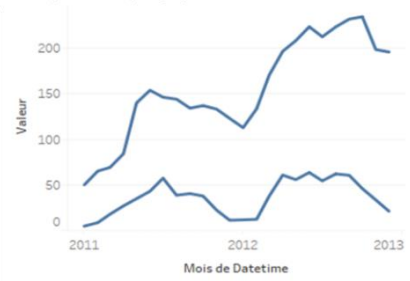



Les statistiques descriptives qui suivent permettent de se familiariser avec les données et de confirmer ou infirmer ces intuitions. (Graphiques effectués avec le logiciel Tableau.)

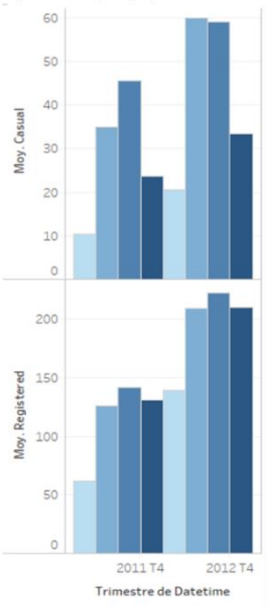
Nombre moyen de locations de vélo par heure en fonction de l'heure de la journée (2011-2012) – graphe 1



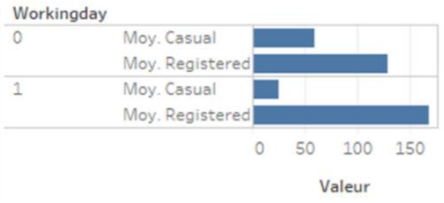

Evolution du nombre moyen de locations de vélo par heure (2011-2012) – graphe 2



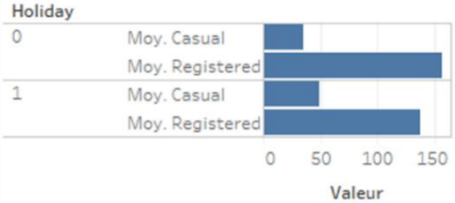

Nombre moyen de locations de vélo par heure en fonction des saisons (2011-2012) – graphe 3



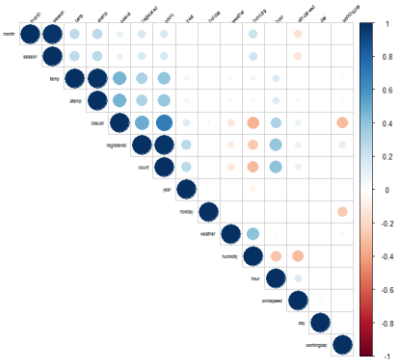

Nombre de vélos moyen loués par heure en fonction des jours travaillés (2011-2012) – graphe 4




Nombre de vélos moyen loués par heure en fonction des vacances scolaires (2011-2012) – graphe 5




Corrélation entre les différentes variables – Graphe 6



Le graphe 1 présente un pic élevé autour de 8h puis 17-18h pour les usagers abonnés contrairement aux usagers non abonnés.
En outre, plus la saison est chaude, plus la demande de vélo augmente mais ceci s'observe d'autant plus fortement pour la demande de vélo des non abonnés (graphe 3).

Le graphe 5 montre que la demande de vélo pour les non abonnés augmente durant les vacances scolaires. A contrario, pour les abonnés, celle-ci baisse. Inversement le graphe 4 montre que la demande de vélo pour les non abonnés diminue les jours travaillés, tandis que celle-ci augmente pour les abonnés.

Le tableau des corrélations (graphe 6) va dans le même sens (nous observons que certaines variables sont corrélées positivement à la demande de vélo des abonnés mais négativement à celle des non abonnés et inversement). Nous constatons aussi que la température et l'humidité sont davantage corrélées avec la demande des non abonnés.

Par ailleurs, nous observons que les variables *temp* et *atemp* sont très corrélées (corrélation de 0,98). Cette corrélation pouvant poser problème pour certains modèles de prédiction, une procédure de sélection de variables sera mise en place. De même pour les variables *month* et *season*.

Le graphe 2 indique une augmentation de la location de vélos durant les années 2011-2012 pour les utilisateurs abonnés ; inversement la demande des non abonnés a été globalement stable. Ceci pourrait illustrer un changement d'habitude de la part des abonnés (développement d'un comportement éco-responsable ?). Nous observons par ailleurs une périodicité au cours des mois/saisons.

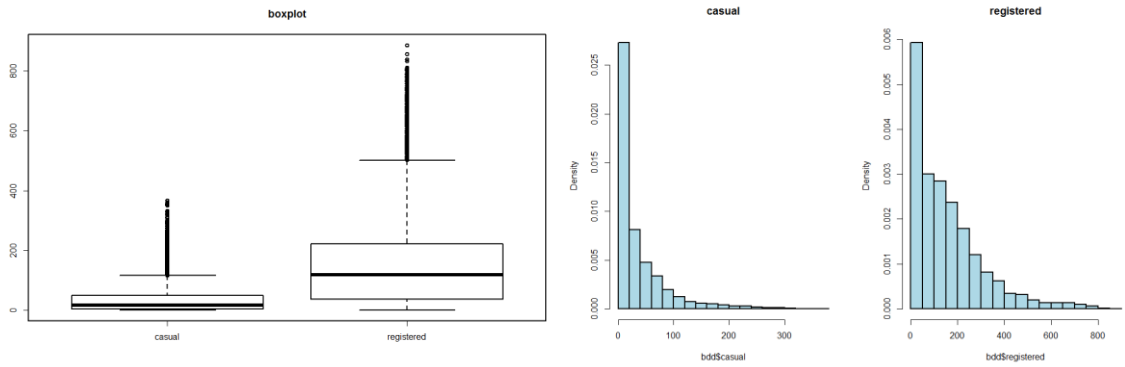
Ces constats confirment les intuitions préalablement présentées. Nous observons deux comportements de demande distincts :

- les abonnés ont tendance à davantage utiliser le vélo pour effectuer les déplacements de type domicile-travail ; leur demande de location dépend donc plutôt de facteurs temporels (jour de travail, heure de la location),

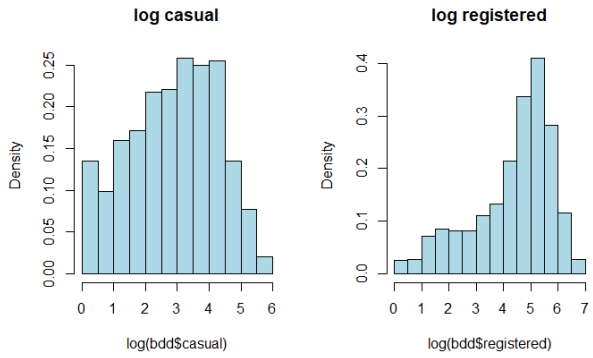
- l'utilisation du vélo des non abonnés s'apparente plutôt à du loisir et est liée à des facteurs tels que la température ou le fait d'être en week-end.

Dans ce contexte, nous décidons d'utiliser **deux modèles distincts** pour ces deux types d'utilisateurs (variables *Casual* et *Registered)*.

Nous observons enfin la distribution des deux variables d'intérêts, *Casual* et *Registered*. Deux constats importants devront être pris en compte lors de la modélisation : ces variables présentent une distribution asymétrique et des valeurs extrêmes.




Par la suite, nous normaliserons ces variables via une transformation logarithmique (cf. graphe ci-dessous) afin de réduire l'asymétrie et de limiter l'influence des valeurs extrêmes.




## 3) La modélisation utilisée

Un prétraitement est effectué sur les données. Les variables catégorielles *year, month* et *hour* sont extraite de la variable *datetime*. Nous créons ensuite des variables binaires à partir des variables suivantes : *season, weather, holiday, workingday, year, month* et *hour*.

Nous avons conçu un **échantillon d'apprentissage** constitué des 14 premiers jours de chaque mois et un **échantillon de test** constitué des jours 15 à 19 de chaque mois. L'échantillon de test permettra de comparer la qualité de généralisation des différents modèles que nous ajustons avec le moins de biais possible (le but ici est de limiter le sur-apprentissage).

Notre variable d'intérêt dénombre le nombre de vélos loués dans un laps de temps donné. C'est pourquoi nous ajustons une régression de poisson adaptée à ce type de données. Cependant nous employons également des méthodes de prédiction de variable continue car la variable d'intérêt (*Registered* ou *Casual)* prend ses valeurs dans un intervalle « large ». Nous ajustons donc successivement une **régression linéaire** et une **régression de poisson**. Pour ces deux modèles de régression (modèle 1 et 2), l'interaction des variables *workingday-hour* d'une part et *month-year* d'autre part sont prises en compte. Nous employons par ailleurs une méthode de sélection de variables en comparant différents modèles au moyen du critère BIC.

Nous utilisons également l'algorithme **kNN** pour prédire la demande de vélo (modèle 3). Afin de déterminer la valeur de k nous utilisons une stratégie « *hold out* » en construisant un sous-échantillon d'apprentissage constitué des 11 premiers jours de chaque mois et un échantillon de validation constitué des jours 11 à 14 de chaque mois. En outre, les données sont normalisées sur un intervalle de longueur 1 afin que les valeurs des différentes variables n'impactent pas le calcul de la distance utilisée dans l'algorithme (distance euclidienne ici).

Nous avons ensuite ajusté un **arbre de régression** pour prédire le nombre de vélos loués (modèle 4), puis afin d'améliorer la qualité de prédiction, nous avons utilisé deux méthodes d'ensemble fondées sur les algorithmes de bagging et de boosting.

Nous avons donc construit un modèle de prédiction fondé sur l'algorithme des **Forêts aléatoires** de Léo Breiman (modèle 5). Cet algorithme constitue une amélioration du bagging dans le cadre des arbres de décision. Comme pour le bagging, les forêts aléatoires permettent de prendre en compte les prévisions de plusieurs arbres, cependant les forêts aléatoires ajoutent une « randomnisation » portant sur le nombre de variables explicatives, rendant les arbres plus indépendants réduisant ainsi la variance du modèle de prédiction agrégé.

Enfin, nous avons utilisé l'algorithme **Gradient BoostingTree** (modèle 6) qui permet de combiner plusieurs prédicteurs. Plus précisément, dans le cadre du gradient boosting, le modèle final est construit pas à pas, en déterminant à chaque étape un nouvel arbre. L'arbre t+1 concentre son apprentissage sur ce qui n'a pas été bien modélisé par l'arbre précédent, en modélisant le résidu de l'arbre t.

L'annexe 1 présente de façon détaillée les méthodes d'ensemble utilisées dans ce rapport.

Hormis pour la régression de Poisson, nous avons ajusté chacun de ces modèles en effectuant une **transformation logarithme** sur la variable de réponse (le nombre de vélo loués) ce qui permet d'obtenir de meilleurs résultats compte tenu de la distribution asymétrique de cette variable.

### 4) Le critère de performance utilisé

Comme indiqué dans le tableau des résultats, nous calculons pour chaque modèle la racine carrée de l'erreur quadratique moyenne (***RMSE***). Ce critère de performance, couramment utilisé, à l'avantage

d'être facilement interprétable. C'est la racine carrée de la moyenne arithmétique des carrés des écarts entre les prévisions et les observations.
Cependant comme nous mesurons des carrés, nous majorons l'importance des « grosses » erreurs. Le critère ***RMSLE***, calculé sur le logarithme des prévisions et des observations, permet de pallier cette caractéristique (étant ainsi moins sensible aux valeurs extrêmes) et présente un autre avantage : il pénalise plus fortement les sous-estimations par rapport aux surestimations. Cette propriété est intéressante dans le cas présent puisque l'objectif est de pouvoir répondre à la demande de vélos. Ainsi nous calculons ces deux mesures pour comparer la performance de chaque modèle.

## 5) L'ajustement des modèles

Nous présentons ici l'ajustement des modèles cherchant à prédire la variable *Casual*. La méthodologie employée étant la même pour la variable *Registered*, nous ne la présentons pas.

### - L'ajustement d'un modèle de régression linéaire

Nous ajustons ainsi un modèle de régression permettant d'obtenir un modèle peu complexe avec de bonnes capacités de prédiction. Deux variables d'interactions ont été retenues : workingday-hour d'une part et month-year d'autre part.

Nous avons réalisé une sélection ascendante des variables. Le critère utilisé est le critère BIC qui a l'intérêt de privilégier les modèles parcimonieux et donc robustes. Aucune variable n'a été supprimée au terme de cette procédure. Les résultats détaillés de cette régression sont présentés en annexe 2.

### - L'ajustement d'un modèle de régression de poisson

La régression de poisson demande de vérifier l'hypothèse forte d'égalité de la moyenne et la variance de la variable à prédire, or un test statistique indique que nos données sont « sur-dispersées » par rapport au modèle de poisson. C'est pourquoi une régression de poisson « sur-dispersée » et une régression binomiale négative ont été ajustées. Les résultats étant peu concluants, nous ne les présentons pas ici.

### - L'ajustement d'un modèle kNN

La validation croisée de type « *hold out* » indique que le nombre optimal de voisins est égal à 11 (détails en annexe 3). La méthode des 11-plus proches voisins livre des résultats proches de la régression linéaire. Toutefois, cette méthode n'exploite pas l'information livrée par les étiquettes observées de la variable d'intérêt ; afin de prendre en compte cette information, nous implémentons un arbre de régression.

### - L'ajustement d'un modèle par arbre de régression

L'arbre de régression implémenté, ainsi que son interprétation sont présentés en annexe 4. L'ajustement d'un arbre de régression présente des résultats moins bons que la régression linéaire. Nous avons regardé si l'élagage de l'arbre permettait d'améliorer les résultats. Comme cela n'était pas le cas, nous avons décidé d'ajouter les deux modèles présentés ci-dessous.

Ces modèles fondés sur les algorithmes du bagging et du boosting agrègent plusieurs arbres afin d'améliorer la qualité des prédictions (résultats davantage robustes).

- L'ajustement d'un modèle de forêt aléatoire

Nous construisons une Forêt aléatoire d'arbres de régression en suivant la méthodologie de Léo Breiman. Nous utilisons les valeurs des paramètres recommandées par Léo Breiman : nous n'avons donc pas limité la taille des arbres et fixé à 1 le nombre minimum d'observations par nœud et le nombre de prédicteurs sélectionnés pour la scission de chaque nœud est fixé à p/3 (p étant le nombre de variables explicatives). Une valeur plus grande de ces deux paramètres assurerait une plus grande robustesse de chaque arbre, mais risquerait d'augmenter la corrélation entre les arbres et de diminuer le gain de l'agrégation.

Le nombre d'arbres agrégés est de 500. Il est préférable de fixer ce paramètre au plus haut, en saturant les contraintes de temps machine. En effet les Forêts aléatoires n'étant pas sujettes au même sur-apprentissage que le boosting, une valeur plus grande est au pire inutile et au mieux bénéfique.

- L'ajustement d'un modèle par boosting

Nous appliquons alors l'algorithme XGBoost ; celui-ci permet de prendre en compte n'importe quelle fonction de perte. Nous avons choisi la perte quadratique (MSE).

Par son mécanisme adaptatif, le boosting est moins à l'abri du sur-apprentissage que les forêts aléatoires. Ainsi nous jouons sur les paramètres permettant d'éviter le sur-apprentissage en fixant les valeurs des paramètres comme suit : on contrôle la taille des arbres en définissant leur profondeur maximale à 5. Les arbres sont construits sur des échantillons d'observations avec des échantillons de variables. Le taux d'échantillonnage sur les observations est fixé à 0.6 et celui sur les variables est de 0.3. Le paramètre de pénalisation (taux d'apprentissage ou de « shrinkage ») a été fixé à 0.05. La baisse de ce paramètre induit une descente du gradient moins rapide et ainsi la recherche d'une solution présentant moins de risque de sur-apprentissage. Il faut cependant augmenter le nombre d'itérations (ce qui augmente le temps de calcul). Le nombre d'itérations est fixé à 700 et à chaque itération 10 arbres sont générés.

Les paramètres ont ainsi été affinés manuellement. Notons qu'il serait possible d'améliorer la performance du modèle en déterminant les paramètres qui minimisent le critère de performance sur l'échantillon de validation. Il faudrait dans ce cas sélectionner avec précaution les intervalles des différents hyperparamètres pour limiter le temps de calcul. Le graphe en annexe 5 indique la mesure de l'importance des variables.

## 6) Les résultats

Nous présentons dans les tableaux ci-dessous une synthèse des modèles obtenus pour la prédiction de la variable *Casual*. Les résultats de la prédiction de la variable *Registered* sont présentés en annexe 6. Ces tableaux permettent de comparer le pouvoir prédictif de chaque modèle sur l'échantillon de test.

Ci-dessous le tableau recensant le RMSE des différents modèles.

| | Modèle 1 | Modèle 2 | Modèle 3 | Modèle 4 | Modèle 5 | Modèle 6 |
|---|---|---|---|---|---|---|
| | Régression linéaire | Régression de poisson | KNN | Arbre | Random Forest | GBT |
| Variables explicatives / Variable prédite | Ajout des variables d'interaction et sélection de variables | | L'ensemble des variables explicatives sont intégrées | | | |
| casual | 26.45 | 61.08 | 28.74 | 34.25 | 27.86 | 24.53 |

Ci-dessous le tableau recensant le RMSLE des différents modèles.

| | Modèle 1 | Modèle 2 | Modèle 3 | Modèle 4 | Modèle 5 | Modèle 6 |
|---|---|---|---|---|---|---|
| | Régression linéaire | Régression de poisson | KNN | Arbre | Random Forest | GBT |
| Variables explicatives / Variable prédite | Ajout des variables d'interaction et sélection de variables | | L'ensemble des variables explicatives sont intégrées | | | |
| casual | 0.61 | 1.89 | 0.65 | 0.75 | 0.59 | 0.56 |

Rappelons que nous avons ajusté les modèles (à l'exception de la régression de Poisson) sur le logarithme de la variable à prédire afin de transformer la distribution asymétrique de celle-ci en une distribution plus proche de celle d'une loi normale et de diminuer la variance.

Il semblerait que la régression de poisson, présentant un RMSE et un RMSLE très élevés, n'est pas concluante dans le cas présent. L'arbre de régression présente également une erreur supérieure à celle des autres modèles étudiés.

Les méthodes d'ensemble appliquées ici ont permis d'améliorer le pouvoir prédictif de l'arbre de régression. Nous trouvons un RMSE en test de 27.9 avec la forêt aléatoire d'arbres de régression. La performance des forêts aléatoires résulte de la randomisation sur les variables (paramétrage dont les avantages se prolongent dans le boosting). Le Gradient boosting, en concentrant l'effort d'apprentissage sur les observations difficiles à modéliser, réduit encore le RMSE sur le test qui vaut 24.5. L'affinage des paramètres du modèle a permis de diminuer fortement les erreurs du modèle. Ainsi, il semblerait finalement que ce modèle est le plus précis pour prédire variable *Casual*. Cependant nous pouvons noter que la régression linéaire a pour avantage d'être plus lisible (moins « boîte noire ») que ce modèle et n'est que faiblement moins performant (RMSE = 26,45 et RMSLE = 0,61). Aussi, le coût algorithmique de la régression linéaire est moindre.

Afin d'améliorer les différents modèles, nous pouvons également avoir recours à des méthodes de *feature engineering.* Il s'agit, par exemple, d'utiliser des arbres de décisions afin de créer de nouvelles variables pour notre modèle à partir des variables catégorielles en rassemblant certaines modalités. Enfin, il serait possible de procéder à un *stakking* afin d'agréger les différents modèles et améliorer sensiblement la qualité de prédiction.

# Annexes

## Annexe 1 : Les méthodes d'ensemble employées

Nous présentons ici, l'algorithme des Forêts aléatoires de Breiman (Randomforest-RI) et l'algorithme Gradient treeboosting de Chen et Guestrin (Extreme gradient boosting). Ces deux modèles sont fondés sur l'agrégation d'un grand nombre d'arbres CART (ClAssification and RegressionTree).

Mathématiquement, nous pouvons écrire le modèle sous la forme suivante :

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

où K est le nombre d'arbres, $f_k$ est une fonction dans l'espace fonctionnel F, et F est l'ensemble de tous les arbres CART possibles. Par conséquent, notre fonction objectif à optimiser peut être écrite comme suit :

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

avec Ω(f), un terme de régularisation traduisant la complexité de l'arbre.

Les deux algorithmes diffèrent dans la construction des arbres. L'algorithme des Forêt aléatoires repose sur une construction aléatoire d'un ensemble d'arbres tandis que l'algorithme Extreme gradient boosting repose sur une construction adaptative d'un ensemble d'arbres.

- L'algorithme des Forêts Aléatoires

L'algorithme des Forêts aléatoires constitue une amélioration du bagging dans le cadre des arbres de décision. Le bagging (ou bootstrapaggregating), consiste à générer m échantillons bootstrap à partir d'un échantillon d'apprentissage, puis ajuster m modèles sur ces échantillons. On détermine alors la prédiction pour une nouvelle observation en calculant la moyenne des prédictions sur les m modèles.

L'objectif des Forêts aléatoires est de rendre les m modèles ajustés plus indépendants en ajoutant du hasard dans le choix des variables qui interviennent afin de réduire la variance de l'estimateur. L'intérêt est le suivant : considérons Q variables aléatoires $(X_i)_i$ i.i.d chacune de variance $s^2$. La variance de la moyenne de ces variables aléatoires est alors $V(\bar{X}) = \frac{s^2}{Q}$.

Si maintenant ces variables sont identiquement distribuées mais avec une corrélation des variables prises deux à deux, la variance de la moyenne devient alors $V(\bar{X}) = \frac{s^2}{Q} + \frac{(Q-1)\rho s^2}{Q} = \frac{s^2 + (Q-1)\rho s^2}{Q} = \rho s^2 + \frac{(1-\rho)s^2}{Q}$

Comme dans le cas indépendant, le deuxième terme décroît avec $Q$ mais le premier terme limite considérablement l'avantage du bagging. C'est ce qui motive principalement la « randomnisation » de l'algorithme exposé ci-après.

Comme pour le bagging, on construit un nombre important d'arbres fondés sur des échantillons bootstrap des données (i.e. échantillons de taille n obtenus par tirage aléatoire avec remise parmi les n observations initiales). Cependant pour les forêts aléatoires, l'arbre est par ailleurs construit en prenant un ensemble au hasard de variables explicatives avant que chaque nœud ne soit subdivisé. Le tirage à chaque nœud des m variables explicatives se fait, sans remise, et uniformément parmi toutes les variables. Chaque variable a ainsi une probabilité $\frac{1}{p}$ d'être choisie. Cet ajout d'aléa permet

de réduire la corrélation entre les estimateurs et de fait, réduit la variance de l'estimateur agrégé. Le nombre m (m ≤ *p*) est fixé au début de l'algorithme et est donc identique pour tous les arbres. Tout comme pour le bagging, les arbres construits ne sont pas élagués. On calcule la moyenne des prédictions dans le cas de la régression et un vote majoritaire est effectué dans le cas du classement.

- L'algorithme Gradient Boosting

Présentons successivement le principe du boosting, puis le Gradient boosting.

Le boosting cherche à réduire la variance du modèle, comme le bagging, mais aussi le biais de prévision. En effet, comme les arbres sont identiquement distribués par le bagging, l'espérance de B arbres est la même que l'espérance d'un arbre. Cela signifie que le biais d'arbres agrégés par le bagging est le même que celui d'un seul arbre. Ce n'est plus le cas avec le boosting. Le boosting diffère par la façon de construire chaque arbre : chaque arbre est une version adaptative du précédent en donnant plus de poids, lors de l'estimation suivante, aux observations mal ajustées. Intuitivement, cet algorithme concentre donc ses efforts sur les observations les plus difficiles à ajuster tandis que l'agrégation de l'ensemble des arbres permet d'échapper au sur-ajustement.

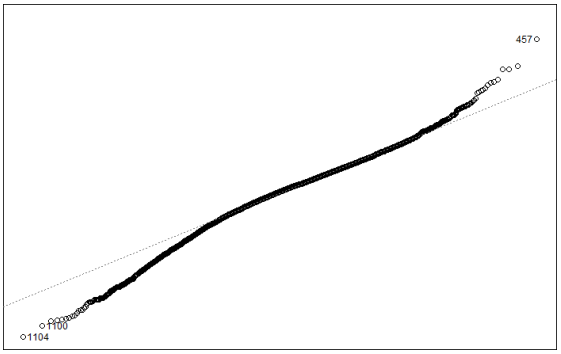
Shapire et Freund sont à l'origine du premier algorithme en la matière, adaBoost (Adaptative boosting). Cet algorithme fut considéré comme une des meilleures solutions sur étagère, c'est-à-dire économisant un long prétraitement des données et un réglage fin de paramètres lors de la procédure d'apprentissage. En revanche, les versions ultérieures, plus performantes en termes de prévision, ont amené à multiplier le nombre de paramètres à régler et optimiser.

Le terme Gradient Boosting vient de l'article « *Greedy Function Approximation: A Gradient Boosting Machine* », de Jerome H. Friedman. L'objectif est toujours de construire une séquence de modèles de sorte que chaque modèle ajouté à la combinaison, apparaisse comme un pas vers une meilleure solution. Mais dans le cas du Gradient boosting ce pas est franchi dans la direction du gradient de la fonction de perte, afin d'améliorer les propriétés de convergence. Nous utiliserons l'algorithme Extreme gradient boosting (XGBoost) de Tianqi Chen et Carlos Guestrin.

## Annexe 2 : Résultats de la régression linéaire

La qualité d'ajustement par un modèle linéaire peut être mesurée grâce au coefficient de détermination ajusté. Ce dernier représente la part de la variance de la variable d'intérêt expliquée parle modèle tout en pénalisant un fort nombre de variables. Ce dernier vaut 0.86, ce qui est satisfaisant. Par ailleurs le test de Fisher nous indique que notre modèle est globalement significatif. En outre, la majorité des variables employées sont significatives au seuil de 1 %.

QQ-Plot des résidus



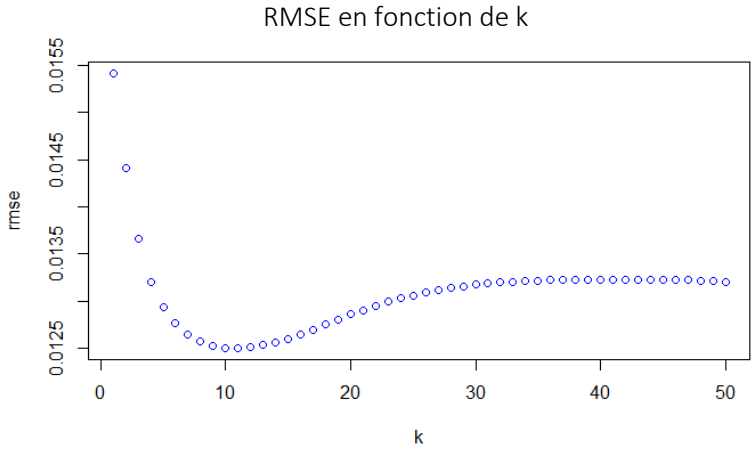
Par ailleurs, l'hypothèse de normalité des résidus, sur laquelle repose le modèle linéaire est vérifiée (cf. graphe ci-dessus).

## Annexe 3 : choix de k pour l'algorithme kNN

Nous utilisons la stratégie de validation croisée « *hold out* ». Plus précisément :

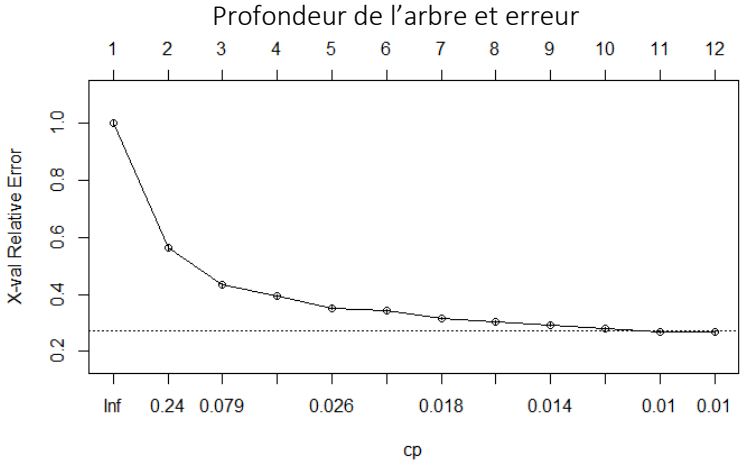
- Nous créons un échantillon de validation, qui regroupe les jours 11 à 14 de chaque mois. Les 11 premiers jours de chaque mois forment la sous base d'apprentissage.
- Pour k allant de 1 à 50, nous implémentons l'algorithme kNN en utilisant la sous base d'apprentissage et les prédictions sont faites sur l'échantillon de validation.
- Nous retenons la valeur de k pour laquelle l'erreur (RMSE) est la plus faible.

Cette démarche nous conduit à retenir **k = 11**.




## Annexe 4 : Arbre de régression

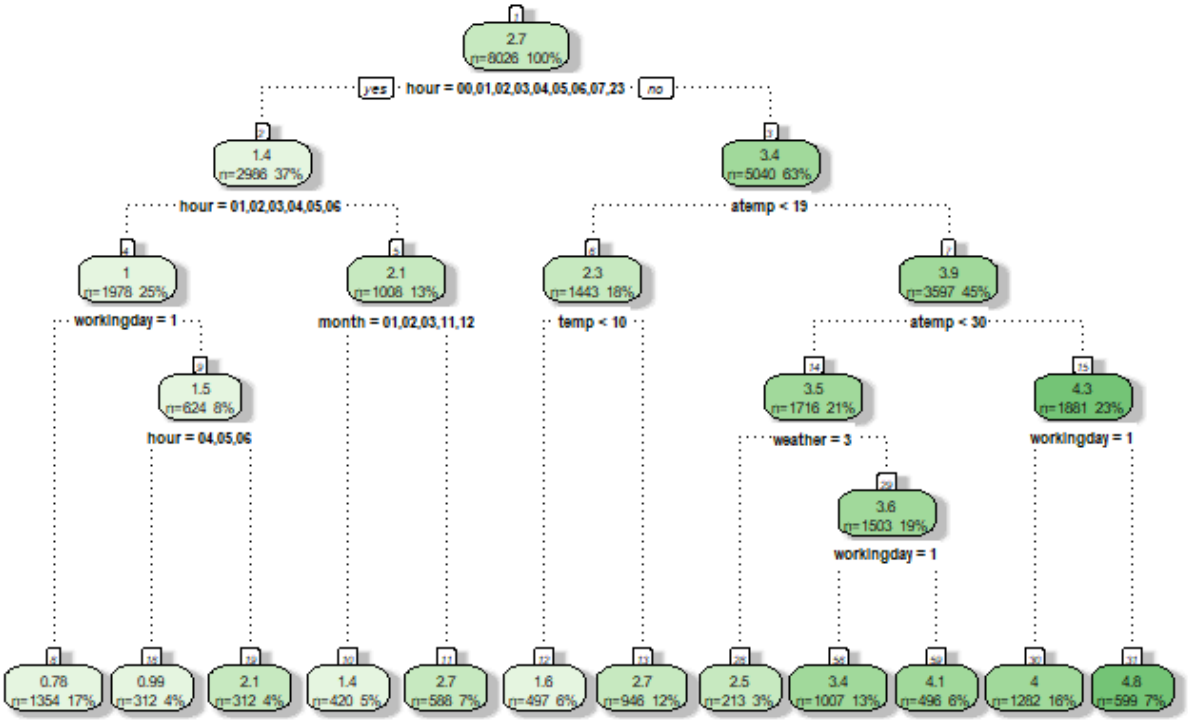
Le critère d'élagage de l'arbre (ou de complexité) est un critère à optimiser afin d'éviter le sur-apprentissage. Le graphe ci-dessous indique que la valeur par défaut de R, cp = 0.01, semble être optimale dans notre cas : l'erreur (calculée par validation croisée en utilisant l'échantillon de validation) est minimisée pour cette valeur (cf. graphe ci-dessous). Ainsi, il n'est pas nécessaire d'élaguer l'arbre.



Par ailleurs, l'avantage principal des arbres est qu'ils sont très facilement interprétables.

Par exemple, grâce à la représentation ci-dessous, on constate que la variable la plus discriminante de la demande de vélo des non-abonnés est l'heure de la location (nuit/jour). En outre, pour les individus qui utilisent le vélo durant la journée (i.e. entre 7 heures et 23 heures), le second facteur influençant le plus significativement la demande de vélo est la température ressentie. Les autres branches de l'arbre peuvent être interprétées de la même façon.

Représentation de l'arbre de régression




Le modèle de prédiction ci-dessus peut être lu très facilement : par exemple, on pourra dire qu'entre 1 heure et 6 heures du matin en semaine (i.e. hors week-end), la demande sera faible et environ égale à 2 (exp(0.78)).

## Annexe 5 : Importance des variables pour le modèle Gradient BoostingTree

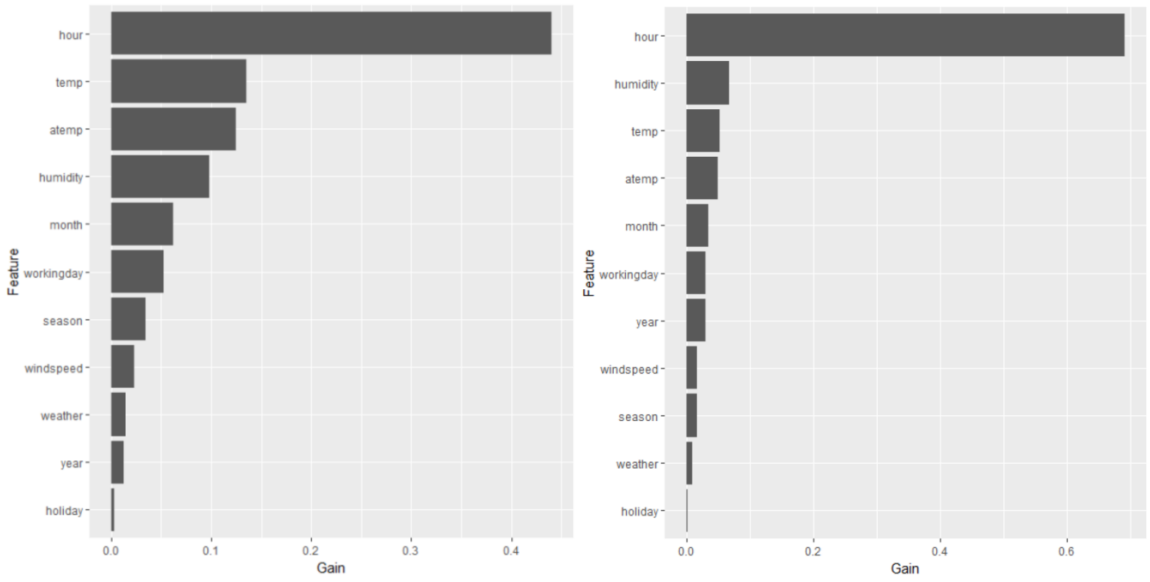
La notion d'importance d'une variable permet de mesurer la contribution d'une variable à la prédiction. L'approche utilisée ici pour déterminer l'importance d'une variable explicative est d'enregistrer la mesure de la décroissance de la variance d'un nœud lorsque qu'une variable donnée est utilisée pour définir la coupure d'un nœud. La somme de ces réductions pour un arbre donné est une mesure de l'importance de cette variable quand cet arbre est construit. On calcule ensuite la moyenne de cette mesure sur l'ensemble des arbres.

On peut noter que Breiman (2001) a suggéré une autre façon d'évaluer le rôle de chaque variable explicative. Celle-ci repose sur une permutation aléatoire des valeurs d'une variable explicative. Plus la

qualité estimée par l'erreur de prévision est dégradée par la permutation des valeurs de cette variable, plus celle-ci est importante.

Les graphes ci-dessous représentent l'importance des variables pour les deux modèles (*Casual* - graphe de gauche et *Registered* - graphe de droite).

Importance des variables




Pour les deux modèles la variable la plus importante est la variable *hour*, mais le gain apporté par cette variable est plus important dans le modèle cherchant à prédire *Registered*. A contrario les variables liés aux conditions météorologiques apportent un gain plus important au modèle cherchant à prédire la variable *Casual*.

## Annexe 6 : Prévision de la variable *Registered*

Nous avons également prédit la variable *Registered* (i.e. la demande de vélo pour les utilisateurs abonnés au service) via une méthodologie similaire à celle présentée précédemment. Nous obtenons des résultats semblables aux précédents (voir tableaux ci-dessous) : la régression linéaire et la méthode du gradient boosting sont les méthodes les plus précises.

| | Modèle 1 | Modèle 2 | Modèle 3 | Modèle 4 | Modèle 5 | Modèle 6 |
|---|---|---|---|---|---|---|
| | Régression linéaire | Régression de poisson | KNN | Arbre | Random Forest | GBT |
| Variables explicatives / Variable prédite | Ajout des variables d'interaction et sélection de variables | | L'ensemble des variables explicatives sont intégrées | | | |
| registered | 47.49 | 215 | 52.42 | 111.41 | 68.67 | 48.48 |

Ci-dessous le tableau recensant le RMSLE des différents modèles.

| | Modèle 1 | Modèle 2 | Modèle 3 | Modèle 4 | Modèle 5 | Modèle 6 |
|---|---|---|---|---|---|---|
| | Régression linéaire | Régression de poisson | KNN | Arbre | Random Forest | GBT |
| Variables explicatives / Variable prédite | Ajout des variables d'interaction et sélection de variables | | L'ensemble des variables explicatives sont intégrées | | | |
| registered | 0.35 | 2.98 | 0.42 | 0.69 | 0.44 | 0.35 |

Notons que les RMSE des différents modèles sont plus élevés que lorsque nous cherchions à prédire la variable *Casual*. Ceci peut être lié au fait que les utilisateurs abonnés sont moins sensibles aux conditions météorologiques. Ainsi nos modèles sont moins efficaces pour prédire *Registered.*

Le fait que le RMSLE soit moins élevé dans certains cas que lorsque nous cherchions à prédire la variable *Casual* peut s'expliquer par le fait que le RMSLE amoindrie l'importance des « grosses » erreurs. Il est probable que nos modèles construits pour prédire *Registered* présente des erreurs plus élevées (demande de vélos moins élastique).

## Annexe 7 : code

Ci-dessus, le code implementé pour *(i)* effectuer les statistiques descriptives et *(ii)* prédire la variable Casual.

```
#_____________________________installation des packages

#install.packages("tree")
#install.packages("useful")
#install.packages("randomForest")
#install.packages("corrplot")
#install.packages("AER")
#install.packages("boot")
#install.packages("MASS")
#install.packages("Metrics")
#install.packages("xgboost")
#install.packages("kknn")
#install.packages('rattle')

library(tree)
library(randomForest)
library(FactoMineR)
library(corrplot)
library(AER)
library(boot)
library(MASS)
library(Metrics)
library(base)
library(xgboost)
library(FNN)
library(kknn)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)


#_____________________________suppression des variables en memoire
rm(list=ls())

#_____________________________importation de la bdd
```

```
BDD<-read.csv("C:/Users/asus/Downloads/data_origine.csv",sep=";",header=T)
#creation des variables day, month, year, hour a partir de la variable datetime
datetime=strptime(as.character(BDD$datetime), format="%d/%m/%Y %H:%M")
day<-format(datetime, format = "%d")
month<-format(datetime, format = "%m")
year<-format(datetime, format = "%Y")
hour<-format(datetime, format = "%H")
bdd<-cbind(day,month,year,hour,BDD[,2:12])


#_____________________________PARTIE 1 statistiques descriptives

#commentaire : certains graphiques ont été tracés sur tableau software

#-----------------------------plot de la série temporelle (une semaine)
plot(datetime[1:168],bdd$count[1:168],type="l")
plot(bdd$count[1:168],type="l")
lines(bdd$casual[1:168],col="red")
lines(bdd$registered[1:168],col="green")

#-----------------------------statistiquesunivariees
names(BDD)
str(BDD)
summary(BDD)
#commentaire : le min pour la variable count est 1, min et médiane diffèrent beaucoup pour
casual, registerd et count
table(day)
table(month)
table(year)
#commentaire : en observant les tableaux de contingence des variables dates nous détectons le
fait qu il y a des données manquantes
table(bdd$season)
table(bdd$holiday)
table(bdd$workingday)
table(bdd$weather)
#commentaire : nous avons une seule ligne présentant la modalité 'fortes averses ou neige', en
majorité 'dégagé à nuageux'

#donnees manquantes
2*12*19*24
#=10944
length(bdd[,1])
10944-length(bdd[,1])
#58 lignes du jeu de donnés sont manquantes
#nous ne mettons pas en place de méthode d'imputation des données car les données manquantes
restent marginales et nous avons seulement des informations sur datetime pour ces lignes

#boxplot casual et registered
boxplot(bdd[,13:14], main="boxplot")
outliersC<-boxplot(bdd[,13], main="boxplot")$out
outliersR<-boxplot(bdd[,14], main="boxplot")$out
#histogramme casual et registered
par(mfrow=c(1,2))
hist(bdd$casual,freq=FALSE,col="lightblue",main="casual",breaks=20)
hist(bdd$registered,freq=FALSE,col="lightblue",main="registered",breaks=20)
par(mfrow=c(1,1))
#log-transformation
par(mfrow=c(1,2))
hist(log(bdd$casual),freq=FALSE,col="lightblue",main="log casual",breaks=20)
hist(log(bdd$registered),freq=FALSE,col="lightblue",main="log registered",breaks=20)
par(mfrow=c(1,1))


#commentaire : ressemble à l'histogramme d'une loi de poisson
hist(rpois(10000,1),breaks=10)

#moyenne,variance casual et registered
mean(bdd$casual)
var(bdd$casual)
mean(bdd$registered)
var(bdd$registered)
#commentaire : une regressionlineaire semble mal adaptee


#boxplot et histogramme variables explicatives continues
hist(bdd$humidity)
```

```
hist(bdd$workingday)
hist(bdd$temp)
hist(bdd$atemp)
hist(data$windspeed)

boxplot(bdd$humidity)
boxplot(bdd$temp)
boxplot(bdd$atemp)
boxplot(data$windspeed)



#-----------------------------statistiquesmultivariees
#correlation entre toutes les variables
library(corrplot)
bdd_cor<-cbind(as.numeric(day),as.numeric(month),as.numeric(year),as.numeric(hour),BDD[,2:12])
mcor<- cor(as.matrix(bdd_cor))
mcor
corrplot(mcor, tl.cex=0.5 ,type="upper", order="hclust", tl.col="black", tl.srt=45)
#Correlation postiveen bleu, negative en rouge
corrplot(mcor, tl.cex=0.3, method = "number", type = "upper")


#boxplot en fonction des modalités des variables factor
boxplot(bdd$count~bdd$year,xlab="year", ylab="count of users")
boxplot(bdd$count~bdd$hour,xlab="hour", ylab="count of users")
boxplot(bdd$count~bdd$day,xlab="day", ylab="count of users")
boxplot(bdd$count~bdd$month,xlab="month", ylab="count of users")
boxplot(bdd$count~bdd$season,xlab="season", ylab="count of users")
boxplot(bdd$count~bdd$weather,xlab="weather", ylab="count of users")
boxplot(bdd$count~bdd$holiday,xlab="holiday", ylab="count of users")
boxplot(bdd$count~bdd$workingday,xlab="workingday", ylab="count of users")

boxplot(bdd$casual~bdd$year,xlab="year", ylab="utilisateur non abonné")
boxplot(bdd$casual~bdd$hour,xlab="hour", ylab="utilisateur non abonné")
boxplot(bdd$casual~bdd$day,xlab="day", ylab="utilisateur non abonné")
boxplot(bdd$casual~bdd$month,xlab="month", ylab="utilisateur non abonné")
boxplot(bdd$casual~bdd$season,xlab="season", ylab="utilisateur non abonné")
boxplot(bdd$casual~bdd$weather,xlab="weather", ylab="utilisateur non abonné")
boxplot(bdd$casual~bdd$holiday,xlab="holiday", ylab="utilisateur non abonné")
boxplot(bdd$casual~bdd$workingday,xlab="workingday", ylab="utilisateur non abonné")

boxplot(bdd$casual~bdd$year,xlab="year", ylab="utilisateur non abonné")
boxplot(bdd$registered~bdd$hour,xlab="hour", ylab="utilisateurabonné")
boxplot(bdd$registered~bdd$day,xlab="day", ylab="utilisateurabonné")
boxplot(bdd$registered~bdd$month,xlab="month", ylab="utilisateurabonné")
boxplot(bdd$registered~bdd$season,xlab="season", ylab="utilisateurabonné")
boxplot(bdd$registered~bdd$weather,xlab="weather", ylab="utilisateurabonné")
boxplot(bdd$registered~bdd$holiday,xlab="holiday", ylab="utilisateurabonné")
boxplot(bdd$registered~bdd$workingday,xlab="workingday", ylab="utilisateurabonné")



#_____________________________PARTIE 2 MACHINE LEARNING

#definition des variables 'factor'
bdd$season=as.factor(bdd$season)
bdd$weather=as.factor(bdd$weather)
bdd$holiday=as.factor(bdd$holiday)
bdd$workingday=as.factor(bdd$workingday)

bdd$casual<-log(bdd$casual+1)
bdd$registered<-log(bdd$registered+1)
bdd$count<-log(bdd$count+1)


#_____________________________definition de l'échantillon d'apprentissage et de test
bdd$day<-as.numeric(bdd$day)
# base d'apprentissage et de test (test du modèle pour éviter le sur-apprentissage)
train<-subset(bdd,day<=14)
test<-subset(bdd,day>15)
# seconde base d'apprentissage et de validation (validation du modèle pour choisir les
paramètres)
subtrain<-subset(train,day<=11)
val<-subset(train,day>11)
```

```
#_____________________________definition bdd

#définition de la bdd pour le modele casual
datC<-train[,c(2,3,4,5,6,7,8,9,10,11,12,13)]
subtrainC<-subtrain[,c(2,3,4,5,6,7,8,9,10,11,12,13)]
valC<-val[,c(2,3,4,5,6,7,8,9,10,11,12,13)]
testC<-test[,c(2,3,4,5,6,7,8,9,10,11,12,13)]


#_____________________________modele1 : regression lineaire

#_____________________________modele pour casual

reg1<-
lm(casual~month+year+hour+holiday+workingday+weather+temp+atemp+humidity+windspeed+month*year+
hour*workingday,datC)

reg2<-step(reg1,k=log(length(datC[1,]))) #procedure de selection de variable par comparaison
du criterebic sur reg 1
                                         #aucune variable n'est supprimée

rmse(exp(predict(reg1,testC))-1,exp(testC$casual)-1)
rmsle(exp(predict(reg1,testC))-1,exp(testC$casual)-1)

reg11<-
lm(casual~year+hour+holiday+workingday+weather+atemp+humidity+windspeed+month*year+hour*workin
gday,datC)
rmse(exp(predict(reg11,testC))-1,exp(testC$casual)-1)
rmsle(exp(predict(reg11,testC))-1,exp(testC$casual)-1)

summary(reg1)
anova(reg1)
# diagnostic plots
layout(matrix(c(1,2,3,4),2,2))
plot(reg1, cex = 1 )


#_____________________________modele 2 : knn

# Convertion des variables categorielles en variables binaires
dmy<- dummyVars(" ~ .", data = subtrainC,fullRank = T)
train_transformed<- data.frame(predict(dmy, newdata = subtrainC))
train_transformed2 <- data.frame(predict(dmy, newdata = datC))

dmy<- dummyVars(" ~ .", data = valC,fullRank = T)
val_transformed<- data.frame(predict(dmy, newdata = valC))
test_transformed2 <- data.frame(predict(dmy, newdata = testC))


# calcul de l'erreur sur la base de validation pour k=1:50
rmse = rep(0, 50)
for (k in 1:50){
cvpred_hold_out=  kknn(casual ~ ., train_transformed, val_transformed, na.action = na.omit(),
                               k = k, distance = 2, kernel = "optimal", ykernel = NULL,
scale=TRUE,
                               contrasts = c('unordered' = "contr.dummy", ordered =
"contr.ordinal"))
# misclassification rates for k=1:25
rmse[k] = rmse(val_transformed$casual, cvpred_hold_out$fitted.values)
}
rmse = rmse/50
rmse

k_star = which.min(rmse)
k_star

plot(rmse, xlab = "k" , col="blue", main="RMSE en fonction de k")

# implémentation du modèle avec k* sur la base d'apprentissage
knn = kknn(casual ~ ., train_transformed2, test_transformed2, na.action = na.omit(),
           k = k_star, distance = 2, kernel = "optimal", ykernel = NULL, scale=TRUE,
           contrasts = c('unordered' = "contr.dummy", ordered = "contr.ordinal"))

rmse(exp(knn$fitted.values)-1,exp(test_transformed2$casual)-1)
rmsle(exp(knn$fitted.values)-1,exp(test_transformed2$casual)-1)
```

```
#_____________________________modele3 :arbre de regression
require(useful)

tree0 = rpart(casual ~., data = datC, method = "anova")
fancyRpartPlot(tree0)
rmse(exp(predict(tree0,testC))-1,exp(testC$casual)-1)
rmsle(exp(predict(tree0,testC))-1,exp(testC$casual)-1)

plotcp(tree0)
ptree<- prune(tree0, cp= tree0$cptable[which.min(tree0$cptable[,"xerror"]),"CP"])
#il ne semble pas nécessaire d'élager l'arbre
fancyRpartPlot(ptree, uniform=TRUE, main="Pruned Classification Tree")

#autreméthode
#apprentissage sur le sous échantillon d'apprentissage
tree_model<-tree(casual~.,subtrainC)

#elagage en fonction de l'échantillon de validation
pruned_model<-prune.tree(tree_model,newdata=valC)
plot(pruned_model$size,pruned_model$dev, type="b") #il ne semble pas nécessaire d'élager
l'arbre

#abre pour predire casual
tree_model<-tree(casual~.,datC)
plot (tree_model)
text(tree_model, pretty=0,cex=0.5)

#_____________________________modele 4 algorithme des foretsaleatoires

#random forest

require(randomForest)
Forest<-randomForest(casual~., datC, importance= F, ntree=500)
```

```
rmse(exp(predict(Forest,testC))-1,exp(testC$casual)-1)
rmsle(exp(predict(Forest,testC))-1,exp(testC$casual)-1)


#_____________________________modele5 : gradient boosting

dtrain<- xgb.DMatrix(data = data.matrix(datC[,1:11]), label = datC[,12])
dsubtrain<- xgb.DMatrix(data = data.matrix(subtrainC[,1:11]), label = subtrainC[,12])
dval<- xgb.DMatrix(data = data.matrix(valC[,1:11]), label = valC[,12])
dtest<- xgb.DMatrix(data = data.matrix(testC[,1:11]), label = testC[,12])

#choix des paramètres
smallestError<- 100
for (depth in 1:10) {
  for (rounds in 190:200) {
xgb<- xgboost(data = dsubtrain,max.depth=depth,eta = 0.1,nround=rounds,objective =
"reg:linear", verbose=0)
      predictions <- predict(xgb,dval)
      err <- rmse(valC[,12], predictions)
    if ( err<smallestError) {
smallestError = err
      print(paste(depth,rounds,smallestError))
    }
  }
}

parametres<- list(objective = "reg:linear",
                  booster = "gbtree",
eval_metric = "rmse",
                  eta              = 0.05,
max_depth        = 5,
                  subsample        = 0.6,
colsample_bytree = 0.3,
num_parallel_tree = 10,
max_delta_step = 10)

xgb<- xgboost(data = dtrain, nrounds = 700, verbose = 1, params = parametres)

#importance des variables
importance_matrix = xgb.importance(model = xgb, feature_names = colnames(datC))
```

```
ggplot(data = transform(head(importance_matrix,50), Feature = reorder(Feature, Gain))) +
geom_bar(mapping = aes(x = Feature, y = Gain), stat = "identity") +
coord_flip()

rmse(exp(predict(xgb,dtest))-1,exp(testC$casual)-1)
rmsle(exp(predict(xgb,dtest))-1,exp(testC$casual)-1)



#_____________________________modele 6 : regression de poisson  /!\ ne pas effectuer la
transformation log

#commentaire : nous avons des v.a.r qui dénombrent le nombre d'occurrences dans un laps de
temps donne => une regression de poisson semble adaptée a ce type de probleme
#Notons cependant que mean(count) et var(count) ne sont pas egaux =>overdispersed poisson
regression

poisreg=glm(casual~month+year+hour+holiday+workingday+weather+atemp+humidity+windspeed+month*y
ear+hour*workingday,datC,family=poisson(link=log))

poisreg2<-step(poisreg,k=log(length(datC[1,]))) #procedure de selection de variables par
comparaison du critère aic
#aucune variable n'est supprimée

rmse(predict(poisreg,testC),testC$casual)
f<-function(x) {
for(i in 1:length(x)) { if (x[i]<=0) x[i]<-0 }
  return(x)
}
rmsle(f(predict(poisreg,testC)),testC$casual)

#test overdispersion
dispersiontest(poisreg,trafo=1,alternative="greater") #H0 equidispersionrejettee

#_____________________________'quasipoisson regression'
quasipoisreg<-
glm(casual~month+year+hour+holiday+workingday+weather+atemp+temp+humidity+windspeed+month*year
+hour*workingday,datC, family = quasipoisson)
```

```
rmse(predict(quasipoisreg,testC),testC$casual)
rmsle(predict(quasipoisreg,testC),testC$casual)

#_____________________________ regression binomiale negative pour casual
library(MASS)
negbinreg<-
glm.nb(casual~month+year+hour+holiday+workingday+weather+temp+humidity+windspeed+month*year+ho
ur*workingday,dat)

rmse(predict(quasipoisreg,testC),testC$casual)
rmsle(predict(quasipoisreg,testC),testC$casual)
```