GLO-4027 - Bike Sharing Demand

Données

20 février 2019

Étudiants en programme d'échange international:
*Bastien **CHABAL***
*Corentin **GIRAUD***

Projet Kaggle – Repository GitHub

Sommaire

Sommaire 1

Introduction 2

Découverte générale des données 2

Les attributs 2

Attribut season (saison) 2

Attribut holiday (vacance) 2

Attribut workingday (jour de travail) 3

Les attributs météorologiques 3

Attribut weather (météo) 3

Attributs temp / atemp (température) 3

Attribut humidity 3

Attribut windspeed 3

Attributs casual / registered / count (nombre de locations) 4

Les relations entre les attributs 4

Difficultés rencontrées 5

Difficulté n°1: le formatage de l'attribut datetime 5

Description 5

Solution 5

Difficulté n°2: les attributs fortement corrélés 6

Description 6

Solution 6

Données manquantes 7

Retour sur la planification du projet 7

Introduction

Le projet que nous avons choisi est intitulé Bike Sharing Demand (plateforme Kaggle). On se propose d'analyser les données du système de partage de vélos de la ville de Washington, D.C.

Le système de partage de vélos au sein d'une ville est très simple : plusieurs kiosques sont répartis dans la ville et des personnes (abonnées ou non) peuvent louer un vélo et faire le trajet qu'elles souhaitent jusqu'à un autre kiosque.

Découverte générale des données

Les données sont réparties en deux fichiers *.csv* distincts:

- Le **set d'entraînement** contient deux ans de données (année 2011 et 2012), où sont relevées toutes les heures différentes informations, dont le nombre total de locations. Ce set ne contient que les données des **19 premiers jours de chaque mois**.
- Le **set de test** contient des données exactement similaires au set d'entraînement, mais pour **tous les jours après le 20 du mois inclus**.

Après une première visualisation des données proposées par le set d'entraînement, on constate qu'il contient 10 886 entrées, réparties sur **12 attributs**. Chaque tuple représente les données pour 1 heure. Tous les attributs ont des valeurs, ce qui signifie qu'il n'y aura pas de prétraitement des données afin de corriger **l'intégralité**.

Les attributs

Nous allons présenter chacun des attributs de manière synthétique pour contextualiser le problème et de rendre toutes les données **interprétables**.

Attribut *season* (saison)

Cet attribut représente le quart de l'année durant lequel la location a été effectuée. Les valeurs sont numériques:

Valeur numérique	Date de début	Date de fin
1	1er janvier	31 mars
2	1er avril	30 juin
3	1er juillet	30 septembre
4	1er octobre	31 décembre

Attribut *holiday* (vacance)

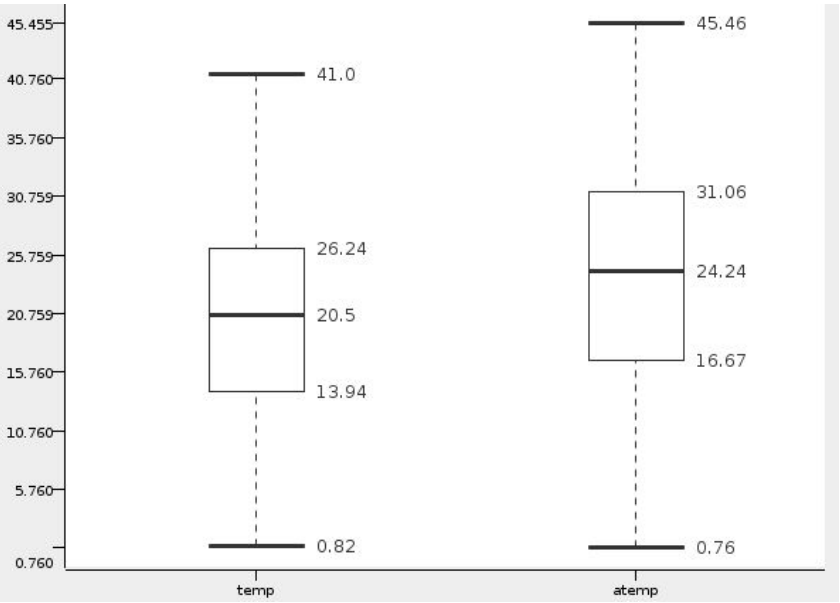
Cet attribut indique si la location a été effectué un jour de vacance. Cet attribut a pour valeur un Boolean (0 ou 1).

Attribut *workingday* (jour de travail)

Cet attribut indique si la location a été effectué un jour de travail. Un jour de travail est un jour non-férié entre lundi et vendredi, hors vacances (cf attribut *holiday*). Cet attribut a pour valeur un Boolean (0 ou 1).

Les attributs météorologiques

Les valeurs de ces attributs sont **exactes** pour tout le jeu de données. En effet, on analysant les diagrammes box-plot plus les valeurs statistiques de chacun de ces attributs, nous n'avons repéré aucune anomalie ou valeur aberrante. On suppose qu'un pré-traitement a déjà eu lieu pour supprimer les erreurs ou le bruit dans les instruments de mesure. Pour illustrer ce propos, voici le diagramme box-plot des attributs de température:



Attribut *weather* (météo)

L'attribut météo possède des valeurs entre 1 et 4 définies comme ceci:

1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Attributs *temp* / *atemp* (température)

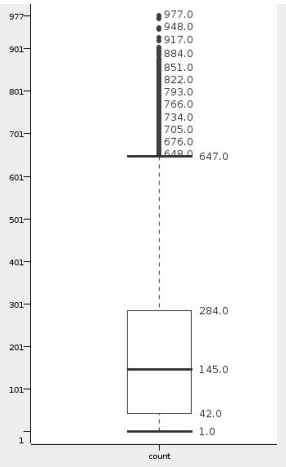
Les attributs *temp* et *atemp* représente la température réelle et ressentie lors de la location en celsius.

Attribut *humidity*

L'attribut *humidity* représente le taux d'humidité lors de la location.

Attribut *windspeed*

L'attribut *windspeed* représente la vitesse du vent lors de la location.



Attributs casual / registered / count (nombre de locations)

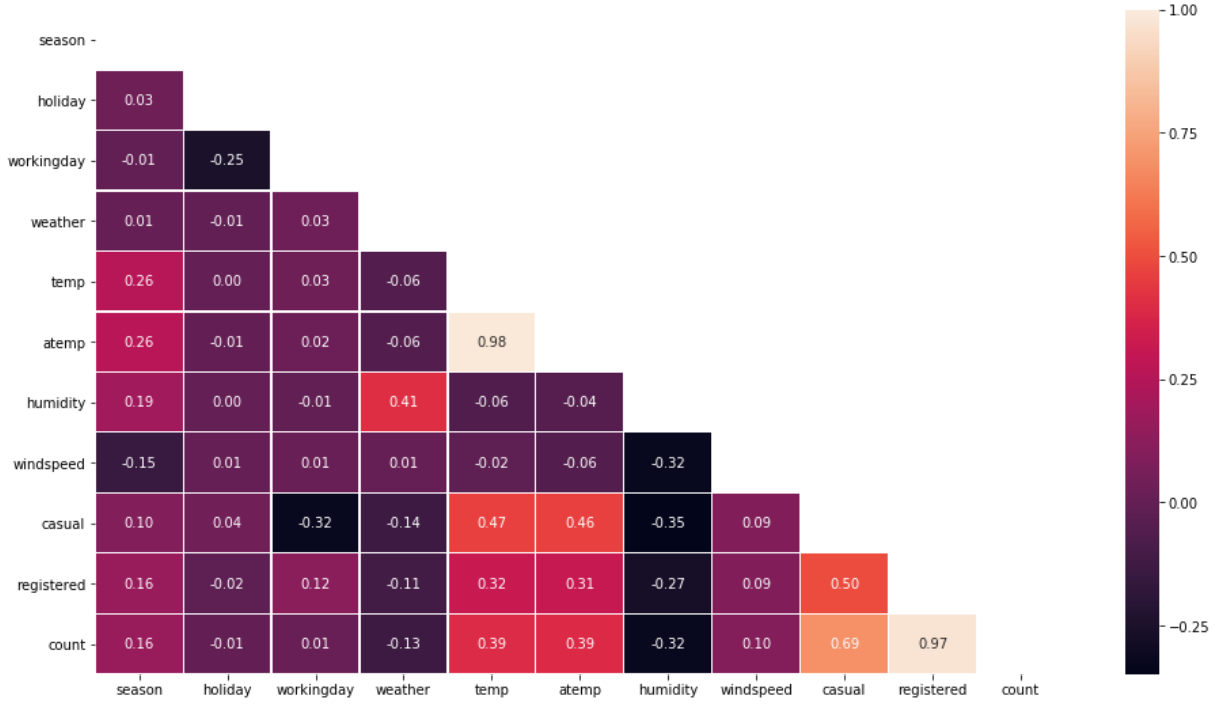
Les attributs casual, registered et count sont très similaires. En effet, casual représente le nombre de locations non enregistrées, l'attribut registered le nombre de locations enregistrées et l'attribut count représente la somme des deux.

Nous avons vérifié que ces attributs sont **cohérents** en vérifiant pour chaque tuple de donnée l'équation suivante :

$$count = registered + casual$$

Les relations entre les attributs

Voici la matrice de corrélation des données :



Avec une corrélation de 0.97, l'attribut *registered* va se comporter de la même manière que l'attribut *bike count*. Ainsi, au lieu d'essayer de prédire directement le *bike count*, une première approche intéressante serait d'abord d'analyser les comportements de *registered* et *casual* séparément. Si ceux-ci s'avèrent différents, on découpe notre problème de base en deux sous-problèmes, amenant à deux prédictions plus précises dont on fera la somme. (Voir Difficulté n°2)

Difficultés rencontrées

Difficulté n°1: le formatage de l'attribut datetime

Description

Le formatage de la date est particulier est n'est pas décliné en plusieurs champs : Heure/Jour/Mois, etc. Cela complique la mise en place d'une prédiction sur un laps de temps plus précis.

- Datetime devrait être séparé en date + time
- Time devrait être un int de 0 à 23 (et non '00:00:00')
- Un attribut devrait être un string du nom du jour (lundi, mardi ...)
- On peut imaginer une autre division : Daypart (nuit, matin, aprem, soirée)

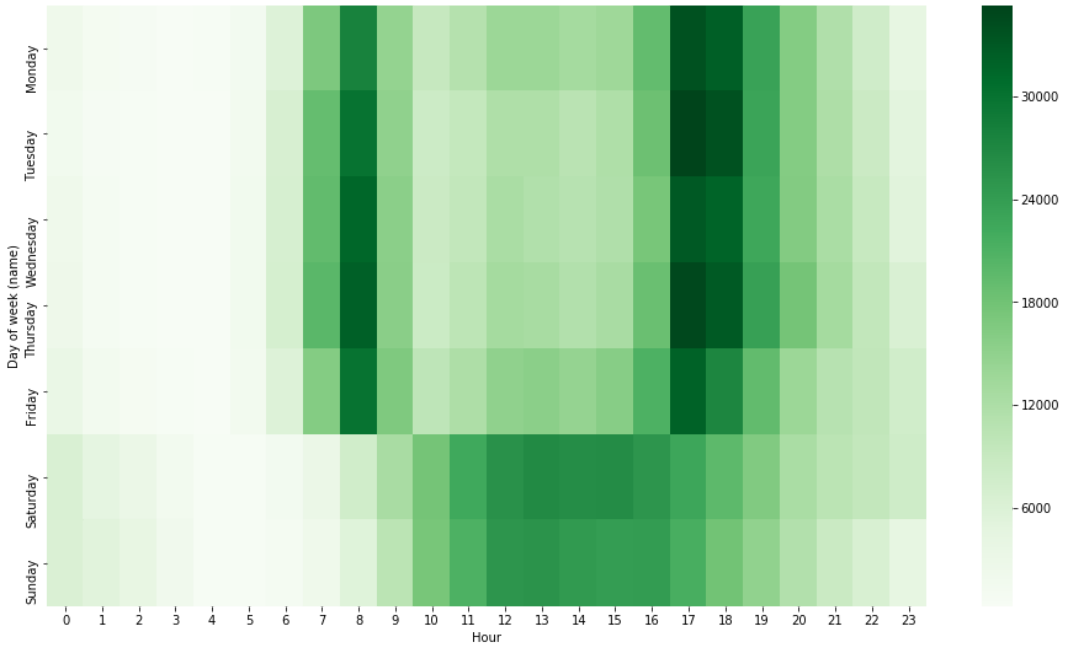
Solution

On peut commencer à implémenter une méthode de transformation des données : la hiérarchie des concepts.

D'abord l'attribut day qui serait une string contenant le nom même du jour : "Lundi", "Mardi", "Mercredi", etc.

Ensuite, on peut diviser ce jour en 24, une division pour chaque heure. L'attribut *hour* serait un int allant de 0 à 23. 0 correspondant à l'heure entre minuit et 1h, 1 à l'heure entre 1h et 2h, etc.

Ainsi, notre prédiction finale se décline en une somme de prédictions plus ciblées et donc plus précises. Cette méthode nous permet de créer la heatmap suivante, représentant la somme des locations agrégées par heure et par jour de la semaine :



On voit clairement que du lundi au vendredi, les heures de forte affluence sont vers 8h du matin et de 16h30 à 18h30. Un distribution bien différente apparaît le week end.

Si l'on suit la même réflexion qui nous a amené à décider de faire deux prédictions différents pour les locations *casual* et *registered*, on peut imaginer ici faire une prédiction pour chaque carré, suivant le fil ci-après: quel jour sommes-nous ? Quelle heure est-il ? Une somme de prédictions sur une heure seulement, en connaissant le jour, la saison, etc… pour chacun, sera plus précise qu'une prédiction globale pour une semaine, un mois, ou plus.

Difficulté n°2: les attributs fortement corrélés

Description

Avec les attributs *registered*, *casual* et *count*, on a trois nombres de locations. Le dernier étant la somme des deux premiers, est-il nécessaire ? S'il ne l'est pas, cela réduit la dimension des données et facilite la tâche aux algorithmes d'apprentissage.

Solution

Comme nous l'avons exprimer dans le rapport de planification, au lieu d'essayer de prédire directement le *bike count*, une première approche intéressante serait d'abord d'analyser les comportements de *registered* et *casual* séparément. Si ceux-ci s'avèrent différents, on découpe notre problème de base en deux sous-problèmes, amenant à deux prédictions plus précises dont on fera la somme. On entame donc une démarche de **réduction de la dimensionnalité**.

Données manquantes

Le projet Bike Sharing Demand est une compétition officielle Kaggle qui a été largement suivi à son lancement. Ainsi, les données sont d'une très grande qualité et nos recherches et visualisations n'ont jusque là montré aucun données manquantes ou aberrantes.

Cependant, à terme, il est possible d'agrémenter les données météorologiques à l'aide d'APIs, et de peut-être améliorer la qualité de notre prédiction. Par exemple grâce à l'API suivante: https://openweathermap.org/history

Retour sur la planification du projet

Nous avons vu dans ce rapport que nous souhaitions découper l'attribut datetime en hiérarchie de concept. Ainsi la solution la plus appropriée semble dans un premier temps de mettre en place un algorithme suivant un arbre de décision.

Références

http://brandonharris.io/kaggle-bike-sharing/
https://www.knime.com/
https://www.kaggle.com/c/bike-sharing-demand/discussion
https://www.researchgate.net/post/When_and_why_do_we_need_data_normalization_in_data_mining_algorithms