Projet de spécialité

KAGGLE CHALLENGE

10 juin 2016

Baillet Valérian
Beaupère Matthias
Fischman Adrien
Meuret Thibault

Table des matières

1 Résumé

Ce document permet de montrer le travail effectué par l'équipe kaggle n5. Il permet d'avoir des informations sur les données utilisées ainsi que les interprétations faites sur ces données. Il permet aussi de recenser les démarches empruntées et les algorithmes utilisés pour pouvoir répondre aux différents challenges kaggle.

2 Objectifs

Nous avons décidé de résoudre différents projets kaggle afin de comprendre le principe de prédiction sur un échantillon de données quand on possède un jeu de données d'entraînement. Cette étude s'inscrit dans le cursus de l'ENSIMAG en constituant une application concrète des cours de fouille de données et de systèmes intelligents. Ce document est composé de deux parties : une partie sur un sujet rapide pour se familiariser avec kaggle (bikesharing) et une partie sur un sujet avec un grand jeu de données (Expedia hotels recommendations).

Première partie
Bikesharing

Objectif Soumission d'une solution au problème kaggle : Bike Sharing Demand

1 Le problème

Résumé du problème On possède les données de circulation des vélos du type des velib parisiens entre différentes stations ainsi que les données météo associées. On cherche à prédire dans quelles quantitée sont loués les vélos, dans le système de location de vélo de la ville de Washigton.

2 Étude des données

Données disponibles On possède les données sur deux ans. Les 19 premiers jours de chaque mois constituent les training set, il s'agit des données permettant de calibrer le modèle. Le reste du mois correspond aux données pour valider le modèle.

On possède une jeu de données contenant des informations différentes selon le types de données :

Données d'entrainement et de test : `train.csv` et `test.csv`

- Jour et heure
- Saison
- Vacances, semaine ou weekend
- Temps qu'il fait
- Températures : ressentie et réelle
- Humidité
- Vitesse du vent

Données d'entrainement uniquement : `train.csv`

- Nombre de vélos loués par des inscrits
- Nombre de vélos loués par des non-inscrits
- Nombre de vélos loués au total

3 Introduction aux modèles

Nous avons mis en place différents modèles prédictifs, en abordant le problème de manière différente à chaque itération. Nous allons détailler les différents raisonnements suivis, puis nous comparerons les résultats.

4 Premier modèle : Régression linéaire

4.1 En R

4.1.1 utilisation de données continues

La toute première solution envisagée était d'effectuer une régression linéaire directement sur les données du problème.

4.1.2 Séparation en 2 modèles : semaine et week-end

Le résultat était loin d'être satisfaisant : après avoir exploré les forums de discussion de Kaggle nous avons vu qu'il était possible de descendre à un score proche 0,7. Nous avons essayé d'identifier ce qui pourrait expliquer le problème de convergence de notre modèle ; après étude graphique des données, nous nous sommes rendus compte qu'il fallait trouver un modèle différent pour la semaine et pour le week-end. Nous avons donc découpé le jeu de données selon la valeur de day et avons effectué une regression linéaire pour chaque ensemble.

4.1.3 Introduction de données discrètes et découpage du problème

4.1.4 Modélisation de Poisson

4.2 Python

5 Deuxième modèle : Random Forest

5.1 Présentation du Random Forest

L'algorithme du Random Forest est très utilisé pour le challenge kaggle. Cet algorithme demande une grande puissance de calcul mais certifie une très bonne prédiction dans bon nombre de cas.

Pour illustrer l'algorithme, nous avons représenté avec la figure 1 un arbre de décision minimaliste. Dans ce cas, on observe l'impact de l'humidité sur les

résultats. Pour chaque nœud de l'arbre, le chiffre supérieur correspond à la moyen des locations par heure et le pourcentage correspond à la proportion de la population considérée.

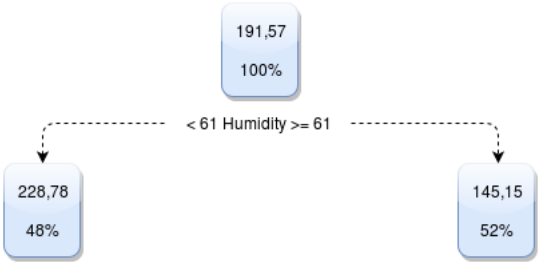

FIGURE 1 – Exemple d'arbre de décision en prenant la variable humidity

Ainsi le random forest correspond à la génération d'arbres de décision dont les variables utilisées sont aléatoires. On fait ensuite une commité de vote entre les différents arbres.

5.2 Python

Nous implémenté cet algorithme en Python. Cela nous a permit d'obtenir un score de 0,68 sur le challenge kaggle. Il est important de préciser que nous avons uniquement pu utiliser 30 arbres alors que cette technique nécessite une centaine d'arbres pour être efficace. Cette limitation est due à un manque de ressources de calcul.

5.2.1 Mise en forme des données

Pour pouvoir appliquer l'algorithme du random forest dans les meilleurs conditions, une mise en forme des données a été nécessaire. Il est à noter que ce conditionnement est similaire à celui employé pour les autres techniques de prédiction. Nous le redonnons ici à titre d'information.

- Nous avons indiquer à notre algorithme de considérer les variable telles que `season` ou `weather` comme des variables discrètes. Ceci est fait en Python grâce à l'instruction `as_type('categorie')`.
- Nous avons créé des nouvelles colonne d'information : la colonne `date` qui contient la date de l'enregistrement au format `Datetime` et qui permet de créer la colonne `hour` qui contient l'heure de la journée sous forme d'un entier.

5.2.2 Algorithme

Le programme Python est donné en annexe . Nous fournissons ici une idée de l'algorithme en pseudo-code.

begin
    Lecture des fichiers csv
    Création de la date et de l'heure
    Discrétisation des variables
    Sélection des données
    Entrainement du modèle
    Prédiction sur le test
    Mise en forme et ecriture du résultat en csv
end

Algorithm 1: Application du Random Forest

5.2.3 Remarque sur le traitement des données

On a ici limité le traitement des données. Ce n'était en effet pas le but premier de cet exercice. Nous cherchions ici plutôt à prendre en main le langage et le forme de challenge kaggle plutôt qu'à optimiser au mieux les algorithme.

Les premières étapes d'un travail plus poussé serait de déterminer de meilleurs variables définissant le systèmes. C'est ce type d'approche que l'on essaiera de développer sur le challenge suivant.

Deuxième partie
Expedia hotels recommandation

Comme deuxième problème, nous avons choisi d'étudier un autre challenge kaggle, plus compliqué cette fois. Plus compliqué par son nombre de variables et par leur complexité. Plus compliqué aussi parce que la quantité de donnée est très importante, soulevant des nouvelles difficultés. En effet, il n'est pas possible de charger toutes les données en mémoire ou même d'effectuer des calculs sur l'ensemble du dataset sans traiter les données et optimiser le code.

1 Explication des données

Pour le problème Expedia, on possède beaucoup de données, on va donc ici détailler les différentes données et leur signification.

1.1 Aperçu des fichiers du challenge

Sur la page du challenge, on nous propose de télécharger quatre fichiers :

- `train.csv` : Regroupe les données d'entrainement. Ce fichier fait 38 millions de lignes. Il s'agit d'un échantillon pris parmi les données d'Expedia des années 2013 et 2014. Ces données correspondent aux recherches d'hôtels et aux réservations faites par les utilisateurs.
- `test.csv` : Regroupe les données de test. Fichier de 2,5 millions de lignes. Ce fichier regroupe les données de 2015 concernant uniquement les réservations faites par les utilisateurs.
- `sample_submission.csv` : Un exemple de fichier à soumettre.
- `destinations.csv` : Des informations sur les hôtels traduis en float (avis sur les hôtels, données de recherche ...)

1.2 Fichiers d'entrainement et de test

Nous étudions ici plus en détail les fichier `train.csv` et `test.csv`. Chaque ligne de ces fichiers correspond à une réservation ou un clic effectué par un utilisateur sur un hôtel lors de sa naviguation sur un des sites internet de Expedia. Nous sont données également les informations sur la recherche qui a conduit à cet évènement.

Donnons tout d'abord le nombre de colonnes de chacun, ainsi que le type des évènements :

- `train.csv` possède 25 colonnes, clic ou réservation
- `test.csv` possède 23 colonnes, réservation uniquement

1.2.1 Contexte de recherche

- `date_time` La date du clic/de la réservation sous forme de timestamp. Attention, `date_time` est relatif au `site_name`
- `site_name` : Identifiant du nom de domaine (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...)
- `posa_continent` Identifiant du continent associé au nom de domaine
- `is_mobile` : Booléen concernant la plateforme d'accès au site. Vaut `1` si l'utilisateur à accédé au site depuis un mobile
- `channel` : Information concernant le moyen d'accès à la page expedia. Par exemple "Direct", "SEM" (via les liens payés à google), "Meta channel" (via Tripadvisor par exemple), etc.
- `date_time` : [Entrainement uniquement] Nombre de clic/réservations effectués pendant la même session par l'utilisateur, les sessions prennent fin après 30 minutes d'inactivité. S'il s'agit d'une session conduisant à un booking `is_booking==1` alors on peut supposer que `cnt==1` car le plus généralement il y a qu'une réservation pas session.

1.2.2 Informations sur l'utilisateur

- `user_location_country` : Identifiant du pays de l'utilisateur
- `user_location_region` : Identifiant de la région de l'utilisateur
- `user_location_city` : Identifiant de la ville de l'utilisateur. Attention, des villes ont le même nom dans différents pays, il faut donc relier `user_location_city` à `user_location_country` systématiquement.
- `user_id` : Identifiant de l'utilisateur

1.2.3 Informations sur l'évènement

- `is_package` : '1' si le clic/la réservation a été généré avec un vol
- `orig_destination_distance‘` : **Distance** physique entre la position de l'utilisateur et celle de l'hôtel cliqué/reservé. Attention : distance exprimée en miles.
- `is_booking‘` : [Entrainement] '1' Si il s'agit d'une réservation, '0' si il s'agit d'un clic

1.2.4 Informations sur la recherche conduisant au clic/à la réservation

- ▷ `srch_cI` : Date de début du voyage spécifié lors de la recherche qui a conduit au clic/à la réservation
- ▷ `srch_co` : Date de fin du voyage
- ▷ `srch_adult_cnt` : Nombre d'adultes faisant partie du voyage
- ▷ `srch_children_cnt` : Nombre d'enfants faisant partie du voyage
- ▷ `srch_rm_cnt` : Nombre de chambre demandé
- ▷ `srch_destination_id` : ID de la destination spécifiée lors de la recherche
- ▷ `srch_destination_type_id` : Le type de cette destination

1.2.5 Informations sur l'hôtel

- ▷ `hotel_continent` : Continent sur lequel se trouve l'hôtel
- ▷ `hotel_country` : Pays dans lequel se trouve l'hôtel
- ▷ `hotel_market` : Zone géographique précise qui recouvre différentes destinations spécifiées dans `srch_destination_id`
- ▷ `hotel_cluster` : [Entrainement seulement] Type d'hôtel, Id du cluster auquel il appartient. Les clusters sont créés en fonctions de la popularité de l'hotel, des notations (étoiles), des avis d'utilisateurs, des prix, des distances aux centre-ville, des aménagements, etc. Les hôtels peuvent appartenir à différents clusters en fonction de la saison (populaire et cher en été pour la plage mais pas cher et pas populaire en hiver par exemple).

1.3 Fichiers de destination

Le fichier `destinations.csv` contient des informations sur les différentes régions recherchées par les utilisateurs. Il contient 62 mille régions différentes et 150 colonnes.

- ▷ La première colonne `srch_destination_id` correspond à une des régions recherchées.
- ▷ Les 149 autres colonnes `d1` à `d149` correspondent à des caractéristiques de la régions basées sur les avis donnés par utilisateurs aux hôtels qui s'y trouvent. Ces informations permettent de construire des similitudes et des différences entre les différentes régions.

Toutes les valeurs de `srch_destination_id` ne peuvent être retrouvées dans destinations.csv. En effet, certaines régions ne possèdent pas d'hôtels assez récents pour construire les informations nécessaire à la complétion de la table des destinations.