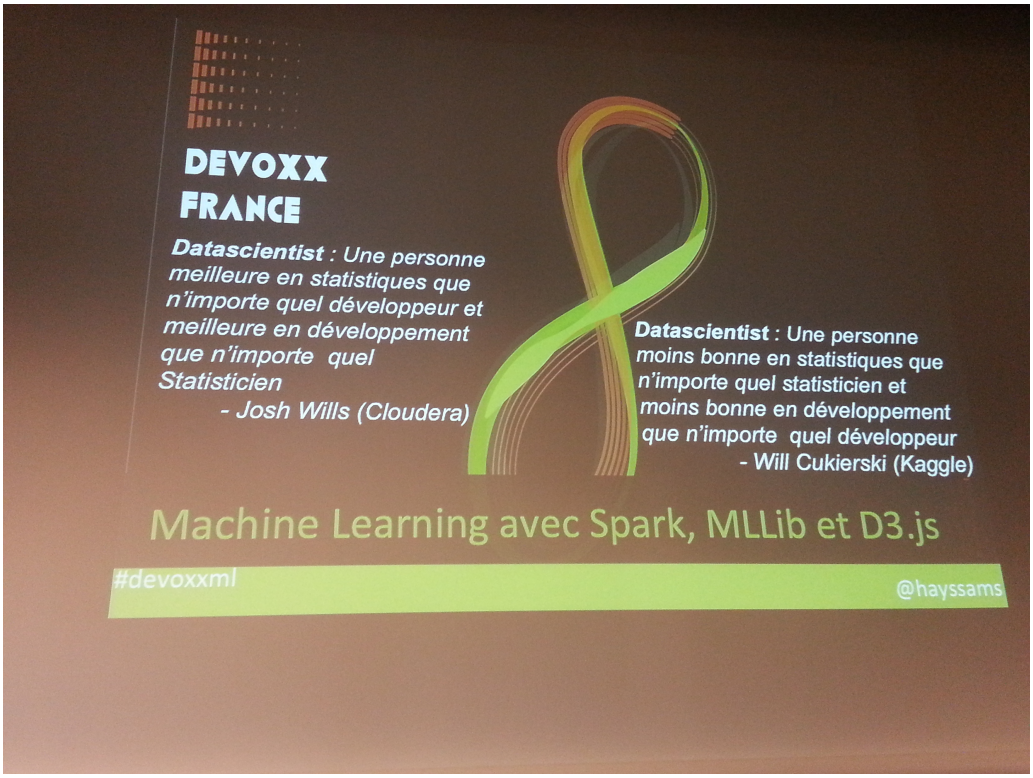
Machine Learning avec Spark, MMLib et D3.js

Date : 9 avril 2015
Format : Conférence
Speakers : Hayssam Saleh, ebiznext

Double objectif : intérêt du machine learning et de Spark.



1. Motivation de l'approche Machine Learning

Le Maching Learning est une branche de l'intelligence artificielle : capacité d'apprendre sans être explicitement programmé.
Il n'y a pas d'algorithme à écrire puisque c'est la machine qui apprend.
Exemple : distinguer les spams

Cas d'utilisation utilisé au cours de la présentation : peut on prêter de l'argent à une personne ?
Prendre des variables indépendantes (les prédicteurs) : salaire, statut marital, propriétaire
En sortie, on a une valeur : OUI ou NON. C'est ce qu'on appelle le **Label**.
Si le nombre de demandeurs grandit, plusieurs solutions sont envisageables :

1. **Engager des analystes** : délai de traitement, cout de traitement, qualité inégale en fonction des analystes.

2. **Système expert à la JBoss Rules** : beaucoup de règles, difficulté de mise à jour du système, devient rapidement opaque, évolution nécessaire en fonction du changement de comportement des clients, évolution en fonction de la population (ex : le salaire dépend de la localisation)

3. **Machine Learning** : on va prendre l'historique des emprunts 2010 à 2015. Choisir ensuite des prédicteurs significatifs. Prendre les résultats (les Labels). On les donne ensuite à manger à un constructeur d'**algorithme de prédiction** (appelé **Modèle**). Avantages :
    a. Précision
    b. Autonome
    c. Performance : une simple formule est à executé
    d. Scalabilité

Moteur de recommandation Pandora : 400 prédicteurs par morceau de musique

Autres cas d'utilisation du ML :
- Recommandation en ligne
- Classification de contenu en groupes prédéfinis (ex : films)
- Regroupement de contenus similaires
- Recherche d'association/patterns dans les actions/comportements
- Détection de fraude et d'anomalies
- Ranking de page

2. Pourquoi le couple Spark / MMLib ?

Avec Spark et MMLib, le Datascientist n'a plus besoin de développeurs.

Dans le monde du Machine Learning, il existe 2 types d'analyses différents :

1. **Analytics d'investigation** : échantillon de données, pas besoin de performance (poste de travail), requête ad hoc offline, métrique : la précision, facilité de développement (language de scripting type R ou Python)
2. **Analytics opérationnelle** : volume de production, cluster de serveurs, sollicitation online continue, métrique : le temps de réponse, performance (JVM Azule)

Spark casse le mur entre ces 2 types d'analyse (datascientist vs développeur)

3. La préparation de l'échantillon des données

Etape cruciale pour avoir des résultats corrects, bien que cela soit l'étape la moins intéressante (un peu comme les tests).

Démarche :

- Prendre un échantillon labélisé contenant l'historique + prédicteurs + labels
- Découper cet échantillon en 2 : donner la 1ière partie au développeur et conserver le 2ième pour tester l'algorithme
- Le développeur va utiliser le générateur d'algorithmes pour créer un **Modèle**
- Le Modèle appliqué à l'échantillon donne la Performance (ex : dans 94% des cas j'obtiens la bonne réponse).
- On applique ensuite le Modèle à la 2nde partie de l'échantillon.

Le modèle est considéré comme satisfaisant lorsque le niveau de précision est considéré comme satisfaisant.

Quelles sont les causes de défaillance du modèle ?

1. Les prédicteurs sont mal choisis : quelque soit l'échantillon, le taux d'erreur est toujours très important (ex : 40% si choix du nom et prénom de l'emprunteur)
2. Overfitting : échantillon non représentatif (ex : que des célibataires dans l'échantillon) ou algorithme de ML non généralisable (variance importance). Des prédicteurs qui n'ont aucun rôle dans la détermination du Label perturbent l'algorithme.
3. Underfitting : l'algorithme de ML est inadapté. Il existe des modèles statistiques et mathématiques. L'utilisation des outils de visualisation permet de choisir l'algorithme approprié.

4. La sélection d'algorithme dans MMLib

MMlib sait résoudre plusieurs types de problématiques :

- **Classification** : Apprentissage supervisé
  - Le résultat est une valeur parmi N sans ordre quelconque
- **Régression** : Apprentisage supervisé
  - Le résultat est une valeur dans un ensemble de valeurs continues. Ex : un nombre
- **Clustering** : Apprentissage non supervisé
  - Les données en entrée ne sont pas labelisés
  - Exemple : comprendre les comportements d'achat
  - Prédire le label de données existantes à partir de leurs prédicteurs. On génère des labels
- **Collaborative filtering**
  - Prédire l'intérêt d'un utilisateur pour un item.
  - Ex : recommandation Amazon
- **Frequent Pattern Matching**

Algorithme utilisé : l'arbre de décision
Recherche du meilleur arbre : recherche de la meilleure condition de segmentation en s'appuyant sur les données.

Les 3 prédicteurs peuvent être modélisés en 3 vecteurs.
L'ordre des règles a son importance pour aller plus vite.
MMLib calcule l'impureté. Pour se faire, il utilise les fonctions Entropy(t), Gini(t), Classification error(t)

Pour cet algorithme, les données doivent être transformées en valeur discrète.
Exemple : salaire < 90 => valeur 1 et salaire > 130 valeur à 4. Attention, 4 n'est pas supérieur à 1. Ce sont des groupements.
Format du fichier :
Label,Salaire,Statut Marital,Propriétaire

La construction de l'algorithme prend en paramètre un niveau de profondeur (maxDepth) qui a un impact sur les performances.

5. La visualisation avec D3.js

La librairie statapss est une surcouche de D3.js
Plusieurs types de diagramme sont disponibles (ex : DensityPlot). Leur utilisation est fonction du type de prédicteur et de label (catégoriel vs numérique).