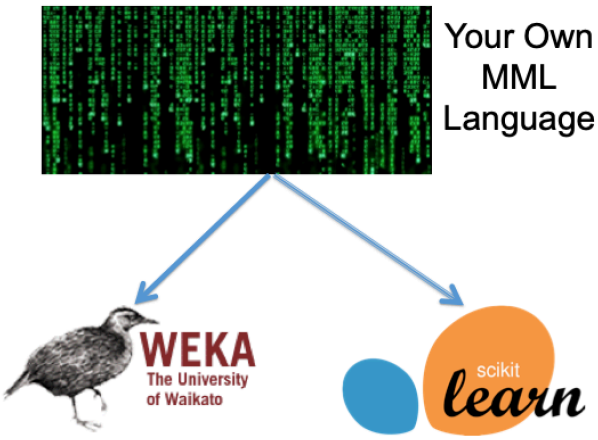
Hack your Own DSLs

TP1+TP2 – Domain-Specific Languages (DSLs), Xtext

On souhaite développer **Multi Machine Learning Language**, un langage textuel, dédié qui permettra de générer/compiler des programmes exécutables, écrits en Python, Java ou R, et permettant d'effectuer des tâches de machine learning (e.g., régression).
Le développement s'effectuera en plusieurs étapes, avec l'utilisation de Xtext.

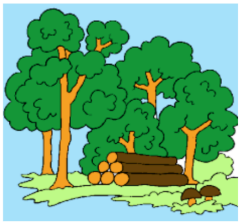
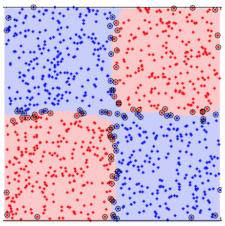
Dans notre contexte l'idée est de pouvoir lancer facilement des campagnes de comparaison entre des mises en oeuvres distincts d'algorithmes de Machine Learning (ML). Ce mini-projet est l'occasion de pratiquer les différents concepts et outils abordés en cours (grammaire, metamodel, compilateur, etc.)



A noter qu'à ce stade nous nous "contentons" d'écrire la syntaxe du langage et nous ne générons/compilons rien du tout… Ce sera le cas au 2ème TP!

Connaissances et techniques acquises à la fin du TP

- Spécification d'une grammaire
- Séparation des préoccupations
- Mise en oeuvre de Xtext pour obtenir une suite d'outils pour un DSL
- Spécification d'instances et utilisation de la syntaxe concrète
- Des difficultés de la conception d'un langage
- Un pas important vers le projet à rendre

Les débuts de Multi Machine Learning Language (MML)

Le but du projet est de travailler sur un outil permettant d'automatiser des tâches de ML. En effet de très nombreuses implémentations de mêmes algorithmes de ML sont maintenant disponibles, dans différents langages (Python, Java, Scala, R, etc.). Nous souhaiterions disposer d'un outil, construit autour de MML, capable de simplifier l'exécution et la comparaison de résultats fournis par ces différentes implémentations.

Nous commencerons en restreignant quelque peu le problème:

- On ne s'intéresse qu'aux algorithmes *supervisés* de *régression* (e.g., regression tree, ordinary linear regression, etc.) https://en.wikipedia.org/wiki/Regression_analysis avec une seule variable cible.
- On prendra en entrée des données au format *CSV* (e.g., le fameux dataset Boston housing)
- On peut paramétrer la stratégie d'évaluation
    - Soit on découpe le jeu de données en deux (training/test) et dans ce cas le pourcentage de données utilisées pour l'apprentissage et le test peut être paramétré (par défaut, 70% training, 30% testing)
    - Soit on utilise la cross-validation (ici encore essayer de fournir les moyens de paramétrer la manière d'effectuer la cross-validation)
- On peut spécifier les variables prédictives et la variable cible
    - Par défaut, toutes les variables sont prédictives sauf la dernière colonne du CSV qui est la variable cible

- En sortie du programme, on souhaite calculer l'erreur de prédiction et différentes métriques existent (MRE, MAPE, etc.).
- On "vise" 2 frameworks de machine learning (pour l'instant):
    - Scikit-learn https://scikit-learn.org/stable/supervised_learning.html
    - Weka https://waikato.github.io/weka-wiki/use_weka_in_your_java_code/

Parmis ces 2 frameworks il y a un large choix d'algorithmes et l'utilisateur pourra mentionner quelle implémentation utiliser par un mot clé.

Le dossier "boston" dans le git du cours donne un exemple avec scikit-learn (exemple basé sur le fameux jeu de données "Boston housing" pour prédire le prix des maisons: https://www.kaggle.com/c/boston-housing)

Vous pouvez jouer avec le jeu de données et l'algorithme de régression via Google Colab:
https://colab.research.google.com/drive/1bF6lEeLwscKiLnkXxlU5-OqGotXNjuMz

Après étude du programme et du jeu de données, il est temps d'écrire un programme, écrit dans votre langage MML, qui contiendra toutes les informations nécessaires pour obtenir boston.py

Q1: Inventer votre propre syntaxe pour le langage MML

Créer une grammaire Xtext de votre langage MML

Créer des exemples représentatifs de l'usage et de l'expressivité de MML

Pour tous vos exemples, écrire le résultat attendu de la "compilation" vers scikit-learn

Commiter/pusher: date limite de rendu, mercredi 16 octobre 23h59 (heure de Paris), l'exercice compte pour 10% de la note de projet

Q2: Présentation des différents langages MML créés par les groupes
Q3: Discussions: mettons-nous d'accord sur le langage!

Former des groupes de 4

- ***remplir le tableur ici https://docs.google.com/spreadsheets/d/1F0rLOFjlECcGBUHMsCTUwglQpEVH-3lyMOaSdpXLCYE/edit?usp=sharing (nom des membres du groupe + adresse du repo sur Github)***