Big Data & Machine Learning

SÉANCE 2 – DASK & TPOT

Organisation

Séance 1
- Introduction sur l'évolution du Big Data et du Machine Learning
- Modélisation avec le Framework Spark (technique de Random Forest)
- Modélisation avec XGBoost (Gradient Boosting)

Séance 2
- Le pourquoi du Framework DASK?
- Utilisation de DASK via une régression logistique et un Gradient Boosting
- Introduction à l'explicabilité / interprétabilité des modèles (techniques Shap et Lime)
- Introduction à TPOT (algo évolutionniste génétique de sélection de modèles)

Séance 3
- Implémentation des techniques de clustering (KMeans, KNN, Spectral)
- Comparaison CPU vs GPU (Framework RAPIDS de Nvidia)

Séance 4
- Distinction des différentes méthodes de réseaux de neurones (Perceptron, CNN, RNN)
- Classification de texte, analyse de sentiment, NLP

DASK

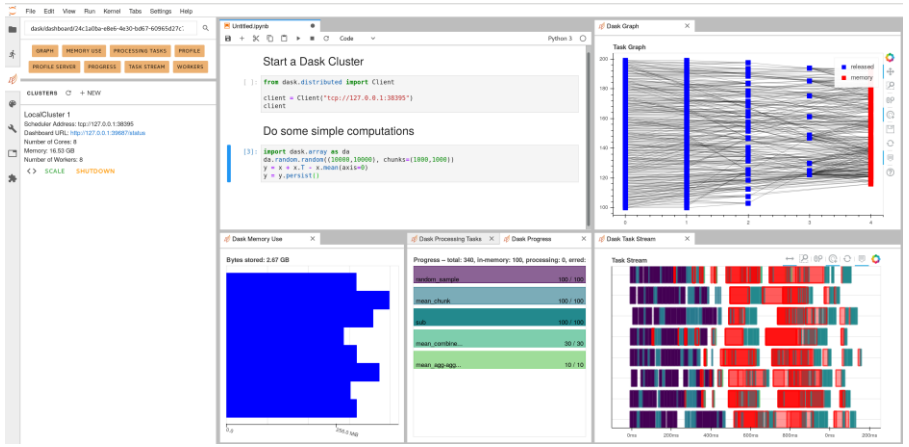
Liens avec son écosystème

Dask

- Interface sur les Frameworks existants : scikit-learn, XGBoost, pandas, numpy
- Même syntaxe, pas d'apprentissage supplémentaire
- Capitalisation sur les communautés existantes, plus de fonctionnalités, meilleure réactivité
- Dask s'intègre à Jupyter pour les consoles de suivi de traitements (dask-labextension)
- Les traitements de ML sont parallélisés et/ou distribués sur plusieurs nœuds (grid search, cross validation…)

Interopérabilité

- RAPIDS (GPU)
- TPOT (AutoML)



DASK

Collections d'objets

- Données **partitionnées** en mémoire et sur les disques au travers de plusieurs neouds
- Dask est **Lazy**
  *df.groupby(xxx).mean()**.compute()***

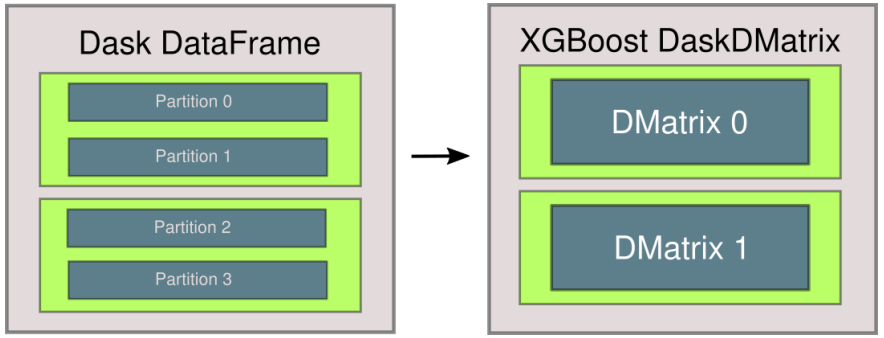
3 types de collections d'objets :

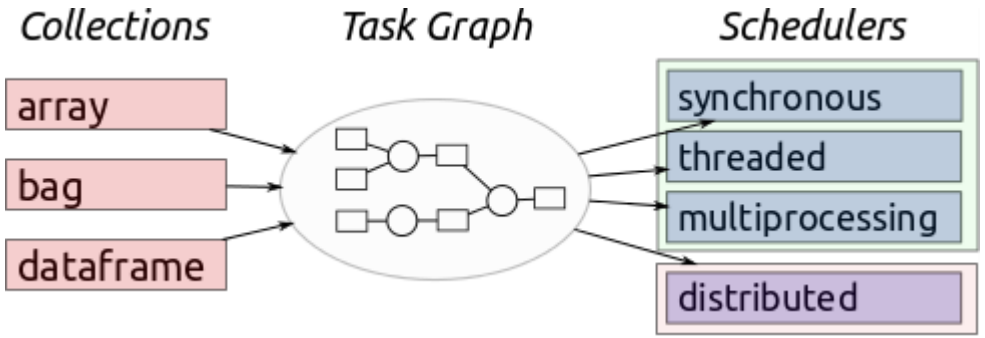
- Arrays : Parallel NumPy
- Bags : Parallel lists
- DataFrames : Parallel Pandas

Le type bag est utilisé pour les données non structurées (json…)

import json

records = b.map(json.loads).filter(lambda d: d["name"] == "Alice")





DASK

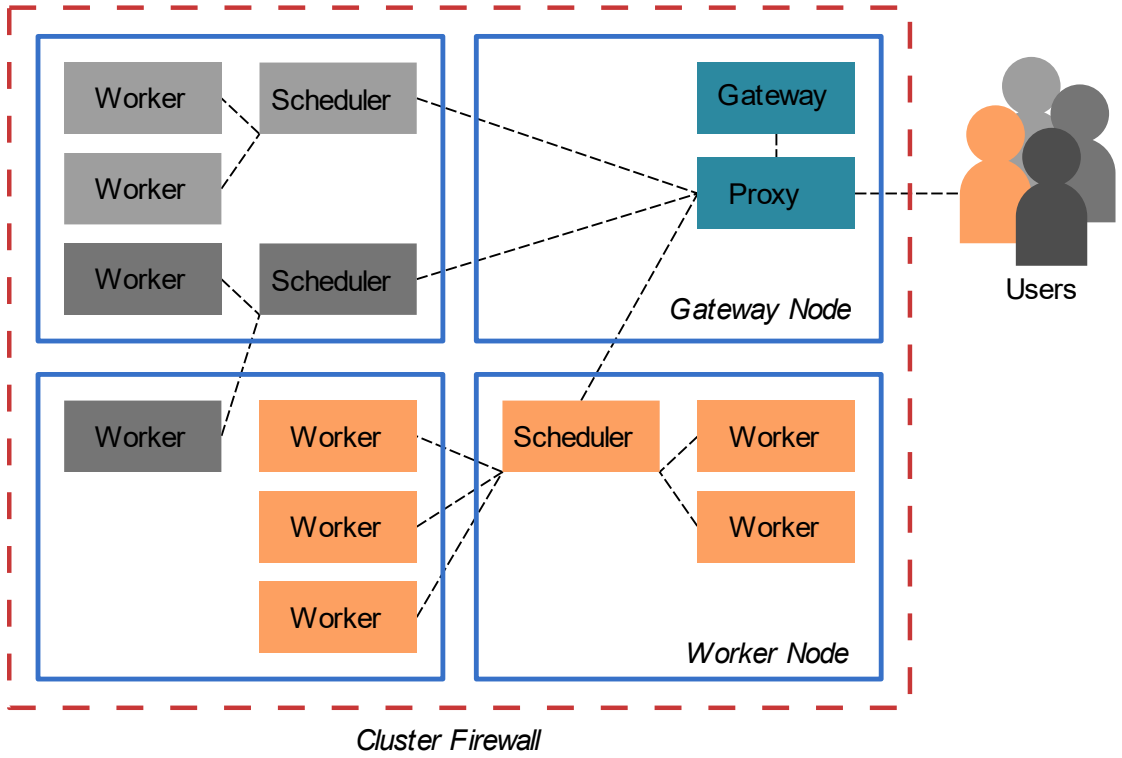
Mode cluster

Cluster YARN

Cluster Kubernetes

- Installation du service dask-gateway sur un edge node, qui permet de lancer des sessions DASK via un port unique
- Pas besoin d'avoir toute la configuration du cluster côté client ou d'ouvrir une large plage de ports

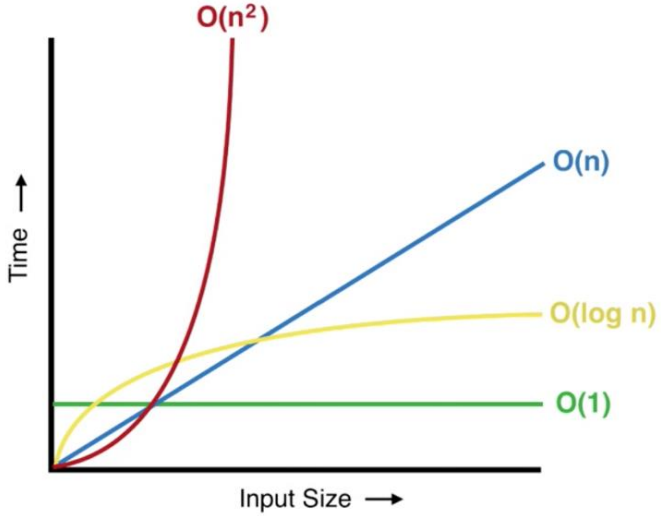
API REST HTTP
Authentification Kerberos possible



2 types de problèmes

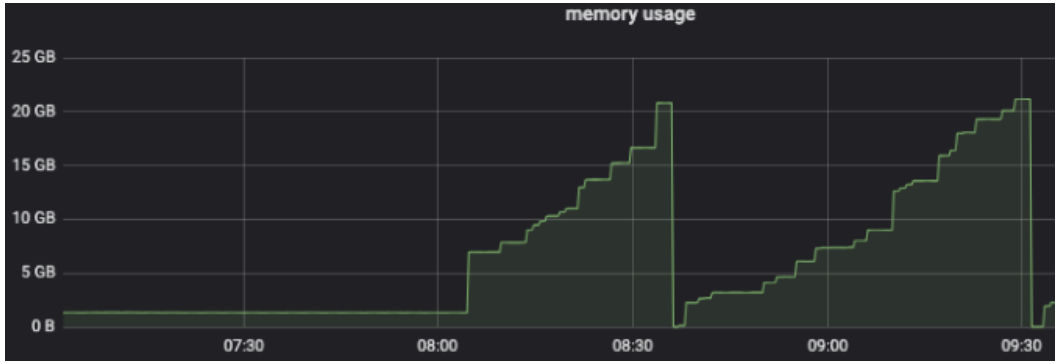
Complexité des algorithmes

Algorithmes longs en temps d'exécution et gourmand en calcul pour converger.
Besoin de parallélisation



La taille des jeux de données

La taille des jeux de données peut ne pas tenir en mémoire sur un seul nœud/serveur, il faut alors distribuer le traitement



TPOT

Algorithme génétique

Exemple d'application avec le problème des 8 dames sur un échiquier : [Lien Notebook Google Colab]

Plusieurs paramètres :

- La population par génération
- Le nombre de génération
- Le taux de mutation (d'un gène)

Fonctionnement

Le but est d'itérer sur plusieurs générations, afin de croiser les individus entre eux pour en créer de nouveaux

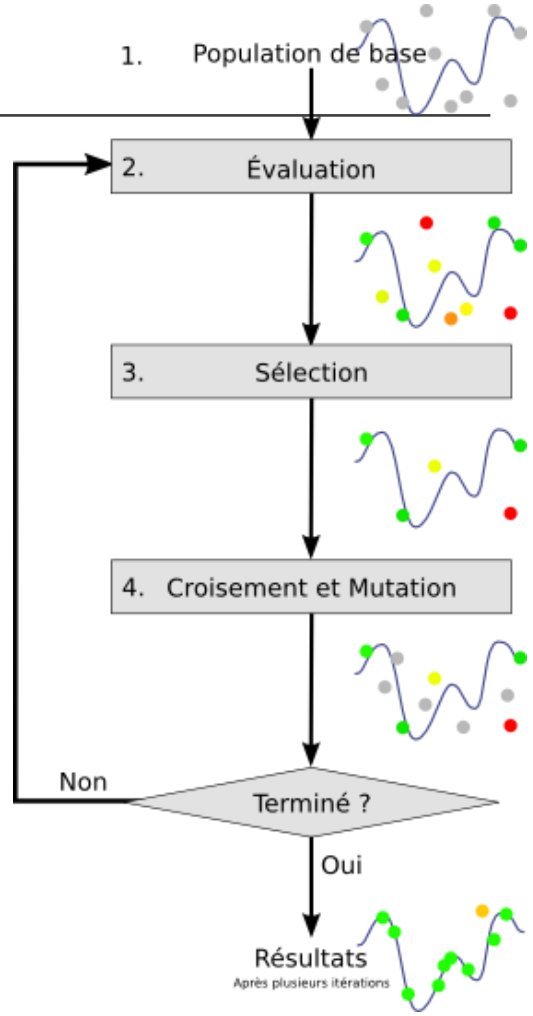
Chaque individu est porteur de plusieurs gènes (qui correspondent à des caractéristiques, paramètres…)

Les nouveaux individus ainsi créés sont le croisement des gènes des deux individus parents

Un taux de mutation très faible peut intervenir pour modifier la valeur d'un gène

Les individus sont sélectionnés pour la reproduction par un système de roue biaisée

- Plus l'individu possède un score proche d'une solution optimale, plus il a de chance d'être tiré au sort
- La fonction de scoring (ou fonction de coût / d'évaluation) score chaque individu, plus il est proche de la solution optimale, plus son score est proche du maximum



TPOT

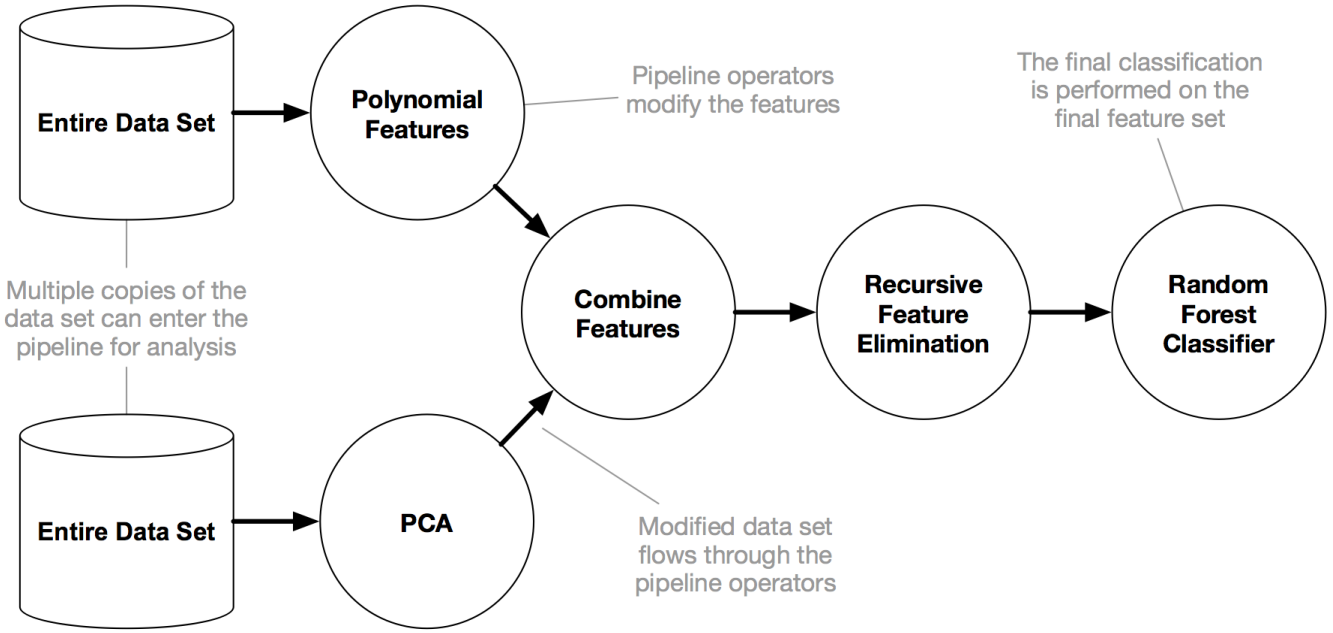
Algorithme génétique

TPOT va donc croiser plusieurs hyperparamètres de plusieurs modèles, pour ensuite les évaluer (fonction de coût), et proposer une ou plusieurs pipelines optimales

[Lien de la doc TPOT]

Pipeline :

- Feature selection
- Preprocessing
- Feature construction
- Model selection
- Hyperparameter tuning
- ~~Data cleaning~~



TPOT

Algorithme génétique

Avantages :

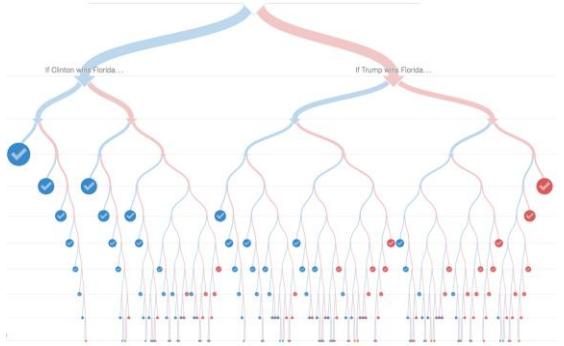
- Plus accessible pour les non initiés via une solution d'AutoML
- Apporte de nouvelles idées de modélisation ou transformation
- Test une grande combinatoire de modèles et de paramètres

Inconvénients :

- Algorithme à complexité exponentielle
  - 20 individus sur 100 générations va générer 2000 modèles
- L'early stopping est toutefois disponible
- Cela doit être utilisé en complément d'une analyse rigoureuse

Explicabilité des modèles

Introduction



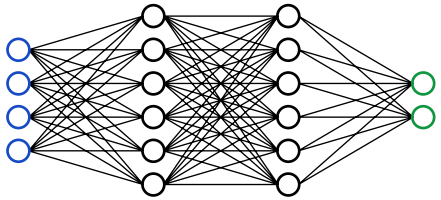
Notebook sur la méthode de permutation importance : lien colab

Les modèles de régression sont facilement interprétable, puisqu'il suffit de se rapporter aux coefficients de chaque variable pour obtenir leurs contributions

Les modèles ensemblistes, ou réseaux de neurones notamment sont beaucoup plus denses, et souvent qualifiés de « black box »

Des méthodes complémentaires sont alors à disposition pour interpréter ou expliquer ces modèles, de manière plus ou moins approximative

- Méthodes pour comprendre le modèle dans sa globalité : MDI, MDA, Permutation Importance…
- Méthodes pour comprendre les prédictions du modèle individuellement : Shapley, Lime…



Explicabilité des modèles

Méthode de Shapley (Shap)

Question : comment chaque feature a contribué à la prédiction comparée à la moyenne des prédictions ? (une valeur continue pour une régression / une proba pour une classification)

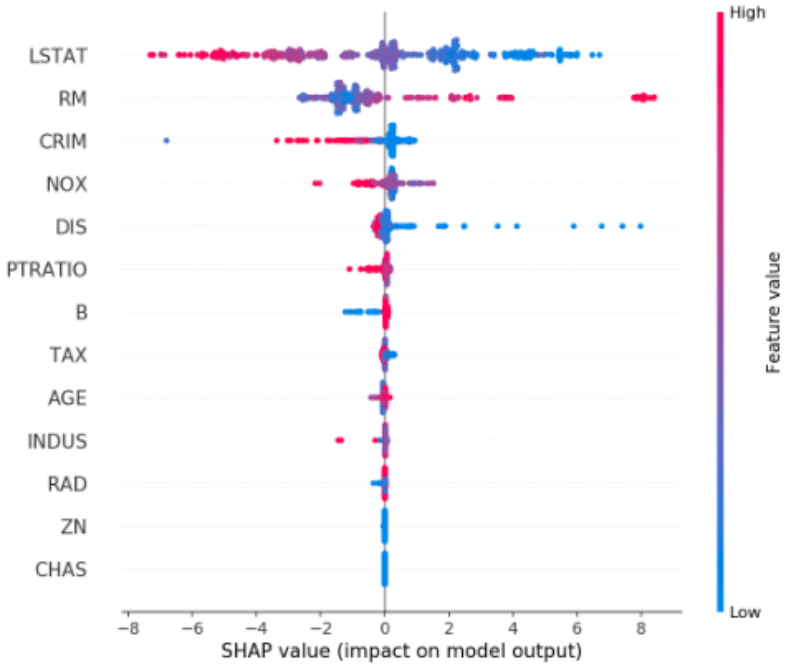
Exemple : Prédiction de vente d'un appart à 300 000, avec une moyenne de 310 000, qu'est-ce qui explique c'est -10 000 ?

Définition : Shapley est la moyenne des contribution marginales d'une feature value à travers toutes les coalitions (combinaisons) possibles.

Fonctionnement : Shapley va permuter aléatoirement les différentes features (étage, emplacement, superficie…), et conserver la différence entre la prédiction simulée et la prédiction réelle. L'opération est répétée pour l'ensemble des coalitions. La Shapley value sera la moyenne de toutes les valeurs de simulation obtenues pour toutes les coalitions/combinaisons possibles, pour chacune des features.

Inconvénient : Temps de traitement long, 2^X coalitions des features value

Avantage : seule méthode avec une théorie solide



Explicabilité des modèles

Méthode Lime

LIME : LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS

Les modèles interprétables : Régressions linéaires généralisées (pénalisées), 1 arbre de décision, K-Nearest Neighbors…

Lime crée un donc un **modèle de substitution** autour de l'observation que l'on souhaite interpréter, Shapley décompose les prédictions finales afin d'obtenir la contribution de chaque feature.
Lime teste ce qui se produit sur les prédictions lorsqu'il fait varier les input features (perturbations légères). LIME génère un nouvel ensemble de données composé d'échantillons permutés, en associant les prédictions correspondantes (du modèle black box), pour entrainer un modèle interprétable dessus. Il va ensuite regarder comment varie les prédictions, en fonction de la variation des inputs. Cela permet de déterminer quels changements de feature impact le plus les prédictions.

Inconvénients :

- LIME montre les influences locales et non globales comme Shapley
- C'est une approximation => non un résultat exact
- Même localement la distribution peut ne pas être linéaire
- De simples perturbations peuvent ne pas être suffisantes
- LIME est dépendant du choix des voisins ou de l'échantillon

Avantages :

- Utilisables sur tous types de données : textes, images, données tabulaires

Lorsque l'on veut interpréter une observation, lime va prendre ses plus proches voisins, pour interpréter les features.

Exercice

MyBinder

Utilisation de la plateforme MyBinder pour l'implémentation de Dask

Spawner une instance à partir du repo Dask : https://github.com/dask/dask-tutorial

Uploader ce notebook

Uploader ce jeu de données