EISTI

Big Data

Généralités sur le Big Data

Maxime LUNDQUIST
Mandry MBUNDU
David RIGAUX
Mehdi DALAA
CPI1 C2

12 juin 2017

Table des matières

Page

1 Introduction **2**

2 Éléments Techniques **4**
2.1 Python 4
2.2 Spark 5

3 Algorithmes Étudiés **7**
3.1 Word Count 7
3.2 K-Means 7
3.3 Text-Mining 8
3.3.1 TF.IDF 9
3.4 Régression Linéaire 10

4 Applications **12**
4.1 Word Count 12
4.2 K-Means 13
4.2.1 Manuel d'Utilisation 14
4.3 TF.IDF 16
4.4 Notes d'informatique 17

5 Conclusion **23**

6 Annexe **26**

1 Introduction

L'an passé, notre projet TIPE portait sur les généralités du Big Data. Notre étude du sujet s'est concrétisée en un rapport et une présentation orale axée en trois parties majeures : les origines du Big Data, les outils informatiques et mathématiques constituant les technologies du Big Data puis les domaines d'applications de ce dernier dans de nombreux secteurs d'activité.

Pour rappel le Big Data désigne un ensemble très volumineux de données qu'aucun outil classique de gestion de bases de données tel que les requêtes SQL ne peut gérer. Bien qu'aucune définition universelle du terme Big Data ne soit reconnue, d'après Gartner, le leader mondial dans les recherches informatiques et de nombreux scientifiques à travers le monde, le Big Data est caractérisé par ce que l'on appelle les 3V. Le premier V pour le Volume considérable de données à traiter, le deuxième V décrit la grande Variété d'informations non structurées provenant de diverses sources, et le troisième V pour la Vélocité c'est à dire la fréquence démesurée à laquelle les données sont générées. Une vision davantage économique introduit un quatrième V décrivant la valeur à extraire par le traitement de ces données. En effet, le marché du Big Data est évalué à 50 milliards de dollars dans les 10 prochaines années.

La Data Science littéralement science des données, est la science qui consiste à collecter, nettoyer et analyser des données hétérogènes pour en extraire de la valeur, tout cela grâce à des outils informatiques et mathématiques. Les notions de Big Data et de Data Science sont donc intrinsèquement liées.

L'une des premières technologies du Big Data sur lesquelles nos études s'étaient portées est MapReduce. C'est un modèle de référence de développement informatique inventé par Google dans lequel sont effectués des traitements en parallèle distribués sur un cluster de centaines de milliers de serveurs. La principale implémentation open source de MapReduce est le framework Hadoop lui aussi l'une des principales et premières technologies du Big Data. Hadoop permet le traitement de données en parallèles de telle sorte que plusieurs machines formant un réseau collaborent à la résolution du même problème. Hadoop est aussi un système de stockage de données. Une autre technologie du Big Data qui a retenu toute notre attention l'an passé et qui nous a servi pour la réalisation du projet notamment dans la programmation d'algorithmes, c'est le framework de traitements Big Data open source nommé Apache Spark, conçu pour effectuer des analyses sophistiquées de données. Performant pour sa facilité d'utilisation et sa rapidité, Spark apporte des améliorations à MapReduce.

MapReduce, Hadoop et Apache Spark sont l'ensemble des outils informatiques du

big data que nous avons étudiés pour réaliser notre projet. En ce qui concerne les mathématiques, elles sont utilisées dans un premier temps dans un processus d'analyse de données. Parmi ces outils on retrouve : le data mining, les statistiques et l'apprentissage. Le data mining est un terme générique englobant tout une famille d'outils facilitant l'exploration et l'analyse de données contenues au sein d'une même base décisionnelle. Les techniques mises en oeuvre lors de l'utilisation de cet instrument d'analyse sont particulièrement efficaces pour extraire des informations depuis de grandes quantités de données. L'apprentissage décrit la répétition d'un processus d'analyse d'un objet par l'étude de différentes variables le caractérisant. D'où la notion de machine learning ou apprentissage automatique qui concerne la conception, l'analyse, le développement et l'implémentation de méthodes permettant à une machine d'évoluer par un processus systématique, et ainsi de remplir des tâches difficiles ou problématiques par des moyens algorithmiques plus classiques. L'analyse peut concerner des graphes, des arbres, ou des courbes.

Notre sujet étant très vaste, nous devions donc nous décider afin d'en tirer une facette exploitable et simulable informatiquement. Nous nous sommes donc orientés vers l'étude d'algorithme de traitement de données. Afin d'être en lien avec le travail effectué en CPI1, les outils que nous allions utiliser pour ce projet devaient être en rapport avec les technologies du Big Data et la data science. Le sujet de notre TIPE porte donc sur les Algorithmes PySpark pour traitement de données. Le traitement de données représente une série de processus qui permettent d'extraire de l'information, du savoir à partir de données brutes. Dans le cadre de notre TIPE les processus étudiés sont des algorithmes programmables. Ils permettent de résoudre différents problèmes de tri en affichant des résultats exploitables par l'humain. Ces résultats peuvent apparaître sous la forme de tableaux, partitions, droites, etc.

C'est grâce à l'interface de programmation applicative (API) d'Apache Spark nommé PySpark que nous avons réalisé la programmation d'algorithme de traitement de données. Cet API nous permet d'utiliser toute la puissance d'Apache Spark avec le langage de programmation Python.

Notre étude sera donc divisée en trois parties : - Les éléments techniques. Dans cette partie, nous décrierons tous les outils informatiques qui ont servi au développement de nos algorithmes. - Les algorithmes étudiés. Dans cette partie, nous expliquerons les principes de fonctionnement des algorithmes de traitement de données que nous avons étudiés. - Applications. C'est dans cette partie que nous étudierons les résultats obtenus grâce aux algorithmes que nous avons étudiés et programmés.

2 Éléments Techniques

2.1 Python

Python est un langage de programmation-objet et multi-plate-forme. Il favorise la programmation fonctionnelle et orientée objet.

Le langage Python fonctionne sur la plupart des plates-formes informatiques, des supercalculateurs aux ordinateurs centraux, de Windows à Unix avec notamment GNU/Linux en passant par macOS, ou encore Android, iOS, et aussi avec Java. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser.

Il est également apprécié par certains pédagogues qui y trouvent un langage où la syntaxe, clairement séparée des mécanismes de bas niveau, permet une initiation aisée aux concepts de base de la programmation.

Python est un langage qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées. Il est cependant particulièrement utilisé comme langage de script pour automatiser des tâches simples, mais fastidieuses, comme un script qui récupérerait la météo sur Internet ou qui s'intégrerait dans un logiciel de conception assistée par ordinateur afin d'automatiser certains enchaînements d'actions répétitives.

Tout au long de notre projet nous avons utilisé Python 3.

Python possède plusieurs modules disponibles pour la création de logiciels avec une interface graphique. Le plus répandu est Tkinter. Ce module convient à beaucoup d'applications et peut être considéré comme suffisant dans la plupart des cas. Néanmoins, d'autres modules ont été créés pour pouvoir lier Python à d'autres biblio-

thèques logicielles, pour davantage de fonctionnalités, pour une meilleure intégration avec le système d'exploitation utilisé, ou simplement pour pouvoir utiliser Python avec sa bibliothèque préférée. En effet, certains programmeurs trouvent l'utilisation de Tkinter plus pénible que d'autres bibliothèques. Ces autres modules ne font pas partie de la bibliothèque standard et doivent donc être obtenus séparément.

Nous avons-nous utilisé la bibliothèque graphique matplotlib.pyplot, bibliothèque graphique fonctionnant comme Matlab. Elle est le résultat d'une association entre pylab, pyplot et NumPy.

NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

2.2 Spark

Spark ou encore Apache Spark est un framework de calcul distribué. Il s'agit d'un ensemble d'outils et de composants logiciels structurés selon une architecture définie. Ce produit est un cadre applicatif de traitements big data pour effectuer des analyses complexes à grande échelle.

Spark fut conçu en 2009 par Matei Zaharia pendant son doctorat à Berkeley. À l'origine, son développement était une solution pour accélérer le traitement des systèmes Hadoop. Les développeurs mettaient notamment en avant la rapidité du produit en termes d'exécution des tâches par rapport à MapReduce. En 2013, Spark devient l'un des projets les plus actifs de la fondation Apache après avoir été transmis à celle-ci. En 2014, Spark a gagné le Daytona GraySort Contest dont l'objectif est de trier 100To de données le plus rapidement possible. Ce record était préalablement détenu par Hadoop. Pour ce faire, Spark a utilisé 206 machines obtenant un temps d'exécution final de 23 minutes alors que Hadoop avait lui utilisé 2100 machines pour un temps d'exécution final de 72 minutes. La puissance de Spark fut démontrée en

étant 3 fois plus rapide et en utilisant approximativement 10 fois moins de machines. Aujourd'hui, il possède plus de 1000 contributeurs.

Spark réalise une lecture des données au niveau du cluster, effectue toutes les opérations d'analyse nécessaires, puis écrit les résultats à ce même niveau. Malgré le fait qu'il s'écrit avec les langages Scala, Java et Python, il utilise au mieux ses capacités avec son langage natif, Scala. Là où MapReduce de Hadoop travaille par étape, Spark peut travailler sur la totalité des données en même temps. Il est donc beaucoup plus rapide pour le traitement en lots et effectuer l'analyse en mémoire. L'API Spark pour Python ou PySpark est le modèle de programmation de Spark adapté au Python.

Spark est divisé en 4 sous applications :

- **Spark SQL**, permet d'exécuter des requêtes en langages SQL pour charger et transformer des données. Le langage SQL est issu des bases de données relationnelles, mais dans Spark, il peut être utilisé pour traiter n'importe quelles données, quel que soit leur format d'origine.
- **Spark Streaming** qui offre à son utilisateur un traitement des données en flux. Il utilise les données en temps réel *DStream*, c'est-à-dire une série continue de RDD.
- **Spark Graph X** qui permet de traiter les informations issues de graphes. *Graph X* étend les RDD de Spark en introduisant le *résilient distributed dataset graph*, un multigraphe orienté avec des propriétés attachées aux nœuds et aux arêtes.
- **Spark MLlib** qui est une bibliothèque d'apprentissage automatique, apparu dans la version 1.2 de Spark, qui contient tous les algorithmes et utilitaires d'apprentissage classiques, comme la classification, la régression, le clustering, le filtrage collaboratif et la réduction de dimensions, en plus des primitives d'optimisation sous-jacentes. On parle souvent d'une fouille de données par apprentissage statistique.

Aujourd'hui, la notion de Big data est très répandue. Spark s'adresse à un public qui recherche un traitement efficace de données trop volumineuses pour être stocké sur un seul ordinateur. Chacun des acteurs concernés recherche de plus en plus de réactivité vis-à-vis du marché d'où l'intérêt d'une analyse en temps réel.

La version de Spark utilisé lors de ce projet était la version 2.10.

3 Algorithmes Étudiés

3.1 Word Count

Wordcount est un programme utilisé pour compter le nombre dans un texte. Il est très utile lorsque l'on traite un texte très long. Il est beaucoup utilisé par les traducteurs pour qu'ils puissent facturer le montant de la traduction rapidement. Il permet aussi de savoir la rapidité de lecture.

3.2 K-Means

K-means ou encore K-moyenne est une méthode de partitionnement des données, c'est-à-dire une méthode de division des données en plusieurs groupes et sous-groupe rangé par caractéristiques à définir. Pour définir k-means, on considère des points et un entier k, notre but est de regrouper ces points en k groupes appelés cluster. Cette application se définit par la formule mathématique :

$$\sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

avec $x = (x_1, ..., x_n)$ un ensemble de n point qu'on cherche à partitionner en k ensemble $S = S_1, S_2, ..., S_n$. Le but de la formule est de minimiser la distance entre les points à l'intérieur de chaque partition. C'est pour cela que sur la photo par exemple, tous les points sont très regroupés. C'est James McQueen qui emploie le terme "k-means" pour la première fois en 1967, mais l'idée venait du mathématicien Hugo Steinhaus en 1957 suite à la diffusion de l'algorithme classique de k-means par Stuart Lloyd la même année. L'algorithme k-means converge toujours à un temps fini, c'est-à-dire qu'il ressort toujours un résultat et que le nombre de partitions est fini selon le nombre k de classes. Celui-ci fonctionne en étape, et à chaque étape de l'algorithme, appelée itération, le centre est de mieux en mieux fixé et l'on a une meilleure partition des points. Le nombre de points présents dans le plan peut faire évoluer de manière exponentielle le temps de calcul c'est pourquoi il est possible d'imposer dès le départ le nombre d'itérations à effectuer, car au final, la solution n'est pas toujours optimale. L'inconvénient avec k-means est que les clusters dépendent de la distance choisie au départ et que l'algorithme doit minimiser la distance entre les points à l'intérieur de chaque cluster. K-means c'est donc :

— Choisir un k constant correspondant au nombre de clusters souhaité
— Initialiser aléatoirement ces k qui seront les centres des clusters

— Calculer pour chaque point le cluster le plus proche pour pouvoir les réunir et organiser les clusters.

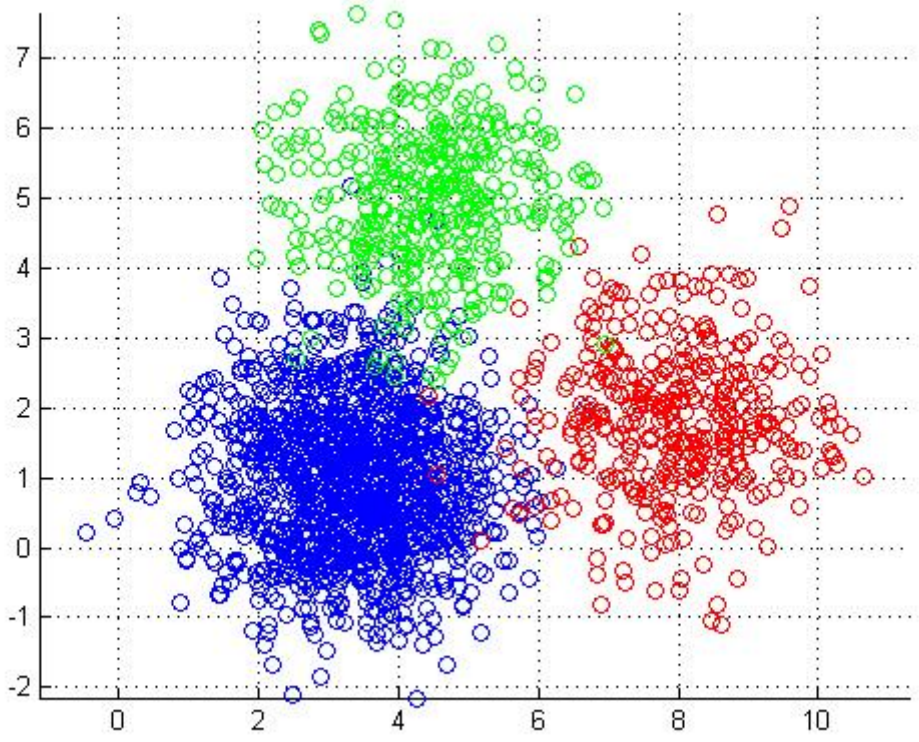

FIGURE 1 – Graphique représentant KMeans

3.3 Text-Mining

Le text-mining ou fouille de textes est une spécialisation de la fouille de données et fait partie de l'intelligence artificielle. C'est-à-dire extraire un maximum d'information des données par les sciences informatiques.

Le text-mining à une méthode de traitement en deux étapes :

— Une étape d'analyse pour reconnaître les différents mots, phrases, leurs relations ou encore leurs sens

— Une étape d'interprétation d'analyse dans laquelle qui nous permettra de choisir un texte plutôt qu'un autre selon nos critères.

Ces deux étapes sont donc dépendantes l'une de l'autre. Le text-mining est donc un procédé dont le but est de classer, structurer ou encore synthétiser un texte en

analysant les relations entre les différents mots. La technique utilisée est le data mining comme déjà dit et nous recherchons à apporter le maximum de connaissance sur un énorme volume de données.

3.3.1 TF.IDF

Nous nous sommes concentrés sur un une application du text-mining dans notre TIPE qui est le TF-IDF ou Term Frequency-Inverse Document Frequency c'est une mesure statistique qui nous permet d'évaluer l'importance d'un contenu par rapport à un document ou à un corpus.

Par définition formelle, nous avons :

— La fréquence brute est simplement le nombre d'occurrences de ce terme dans le document considéré). On peut choisir cette fréquence brute pour exprimer la fréquence d'un terme. Il existe plusieurs manières de l'exprimer que nous représenterons dans le tableau suivant :

Schéma de Pondération	formule du Term Frequency
binaire	$0, 1$
fréquence brute	$f_{t,d}$
normalisation logarithmique	$1 + log\,(f_{t,d})$

— Une étape d'interprétation d'analyse dans laquelle qui nous permettra de choisir un texte plutôt qu'un autre selon nos critères.

— La fréquence inverse de document est une mesure de l'importance du terme dans l'ensemble du corpus. Dans le schéma TF-IDF, elle vise à donner un poids plus important aux termes les moins fréquents, considérés comme plus discriminants. Celui-ci est donné par la formule suivante :

$$idf_i = log\left(\frac{|D|}{|d_i : t_j \in d_j|}\right)$$

avec $|D|$: le nombre total de documents $|d_i : t_j \in d_j|$: nombre de documents où le terme t_i apparaît. Le TF-IDF est alors le produit de ces deux résultats. L'utilisation de cet algorithme est plus développée dans la suite de notre rapport.

Le TF-IDF est utilisé en recherche d'informations, si des documents sont identifiés comme pouvant répondre à une requête, on les ordonne par ordre de pertinence.

3.4 Régression Linéaire

La régression linéaire est un modèle de régression qui cherche à trouver une relation linéaire entre une variable dite expliquée et une ou plusieurs variables dites explicatives. La variable expliquée pour un individu i s'écrit toujours comme combinaison linéaire des variables explicatives :

$$y_i = \beta_0 + \beta_1 x_{1,i} + ... + \beta_k x_{k,i} + \epsilon_i$$

x_i et y_i sont fixes et ϵ_i représente l'erreur.
Le but de la régression linéaire est de prédire un phénomène et de tenter de l'expliquer.
En effet, si nous prenons y comme variable expliquée et x comme variable explicative, nous aurons une droite de y en fonction des valeurs de x, ce qui nous permettra de pourvoir prédire le phénomène. La régression linéaire fonctionne très bien et est très utile dans certains cas tels que la prédiction dans les sciences de l'éducation qui est l'exemple le plus concret.
Nous pouvons également exprimer la variable expliquée pour un individu i vectoriellement, ou en matrice.

Nous pouvons estimer les résultats de la régression linéaires de plusieurs manières différentes :

- La méthode des moindres carrés
- La méthode des moments
- La méthode bayésienne
- La méthode du maximum de vraisemblance

Dans notre cas, nous nous sommes orientés vers la méthode des moindres carrés, qui consiste à minimiser la somme des carrés des distances entre y_i et f(x_i).

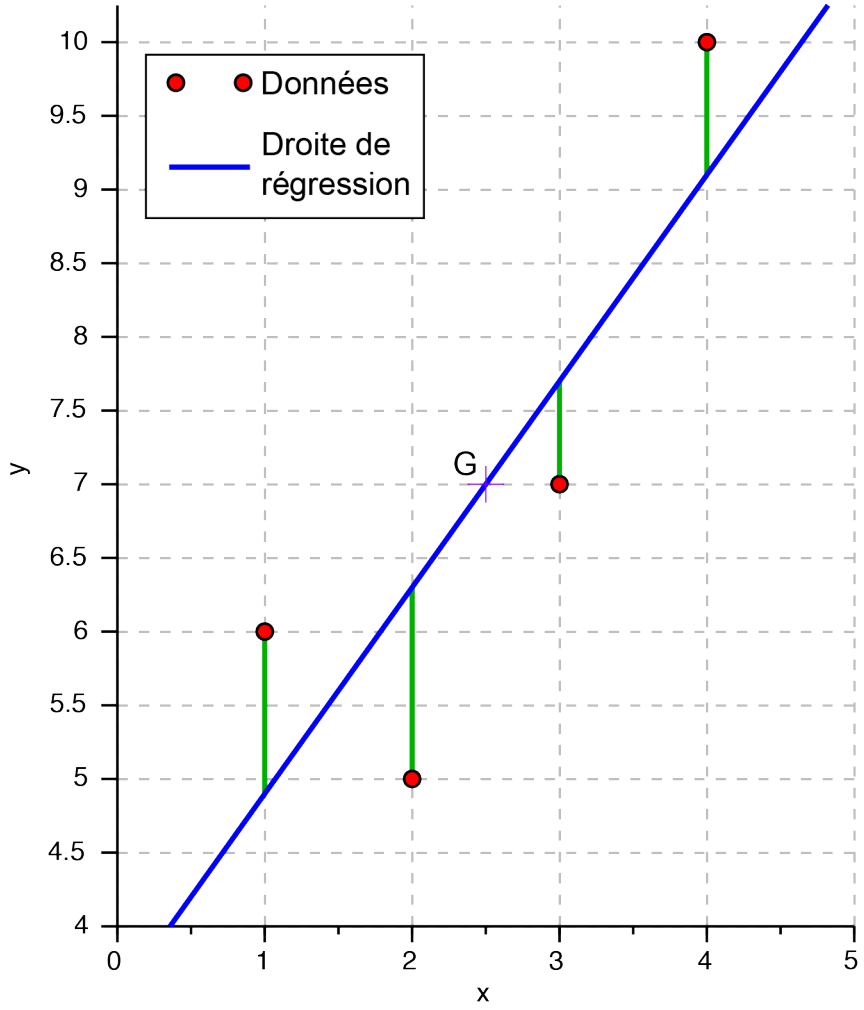

FIGURE 2 – Schéma de la Régression Linéaire

Voici une image représentant une régression linéaire. Nous avons la droite et recherchons donc son équation qui sera de la forme $y = ax + b$. dans cet exemple, nous devons trouver les coefficients a et b. Nous pouvons utiliser ici la méthode des moindres carrés ici dont nous avons parlé dans l'énoncé de notre partie. On a donc bien notre droite de régression avec Y en fonction de X et donc les valeurs de X permettent de connaitre celles de Y. On a donc un modèle de prédiction avec comme objectif de minimiser l'erreur de précision.

4 Applications

Dans cette partie, nous allons parler des différents algorithmes que nous avons réalisé à l'aide de PySpark. Pour pouvoir utiliser les différentes fonctions il faut bien évidemment avoir Spark d'installer sur sa machine. Pour la plus part des fonctions, nous allons faire deux sous parties qui comportent un manuel technique (expliquant comment modifier le code) et un manuel d'utilisation (qui explique comment utiliser la fonction à un utilisateur).

4.1 Word Count

Nous avons implémenté la fonction WordCount, qui nous permet de compter le nombre d'occurrence des mots d'un texte. Le fonctionnement de cette fonction commence par le filtrage du texte à étudier. En effet, nous remplaçons toutes les ponctuations par des espaces, et, sachant que les chaînes de caractères sont sensibles à la cases, nous avons mis l'ensemble du texte en lettres minuscules. Une fois le filtrage effectué, nous séparons tous les mots grâce à la fonction *split()* qui nous permet de séparer les éléments d'une chaîne de caractères avec un certain caractère. Ceci nous permet par la suite d'avoir une liste composé exclusivement des mots, sans ponctuations ni d'espaces. Maintenant, nous couplons chaque éléments de cette liste avec la valeur 1. Il ne nous reste dorénavant, plus qu'à ajouter la valeur numérique du couple avec la valeur numérique d'un autre couple, si la valeur textuel est identique. Cela compte alors le nombre d'occurrence. Pour finir, nous trions la liste en fonction du nombre d'occurrence et ne sortons qu'un certain nombre de mots, ce sont les mots les plus utilisé dans le texte.

Manuel Technique

Si nous souhaitons ajouter ou retirer des filtres pour le texte, il nous suffit de modifier la ligne suivante :

```
1 formattedText = inputFileData.map(lambda x: x.replace("'",' ').replace(
    '.',' ').replace(',',' ').lower()) #every ponctuation transformed +
     lowercased
```

Listing 1 – Modification des filtres

Il faut savoir que cette fonction ressort une liste, donc une connaissance basique des listes en Python peut être nécessaire pour utiliser les résultats ressortis.

Manuel d'Utilisation

Pour utiliser la fonction que nous avons créé il faut tout d'abord ce munir du chemin dans le quel se trouve votre fichier que vous souhaitez utiliser, ainsi que le nombre d'éléments que vous voulez ressortir.

Pour appeler la fonction, il faut tout d'abord modifier certains éléments dans le fichier.

```
1 print (wordcountF(10,"PATH/lipsum.txt"))
```

Listing 2 – Appel de fonction par défaut

Voici l'exemple par défaut dans le fichier *wordcount.py*, nous pouvons remarquer la signature suivante :

```
1 wordcountF(n,chemin))
```

Ceci, nous ressort les n premiers termes les plus répété dans le fichier. *Chemin* correspond en effet à une chaîne de caractère décrivant le chemin du fichier à lire.

Maintenant pour lancer l'application, il faut se rendre dans le dossier principal de *Spark* et lancer dans la console les instructions suivantes :

./bin/spark-submit PATH/wordcount.py

Avec *PATH*, le chemin amenant au fichier *wordcount.py*

```
[(296, u'sed'), (266, u'in'), (213, u'ut'), (201, u'et'), (193, u'sit'), (193, u
'amet'), (190, u'nec'), (189, u'vitae'), (189, u'non'), (183, u'mauris')]
```

FIGURE 3 – Sortie de wordcount avec $n = 10$ sur un texte *Lipsum*

Le premier terme de chaque couple correspond au nombre d'occurrence de chaque mots.

4.2 K-Means

Pour la fonction K-Means, nous avons faits deux versions, une version n'utilisant que les fonctionnalités de bases de Python avec quelques modules rajoutés. Et une deuxième version, qui est implémenté dans l'application Notes d'informatique qu'on verra plus tard dans le rapport. Nous allons dans cette partie, nous concentrer sur la première version. Pour utiliser cette fonction, il faut que les modules *maplotlib* et numpy soit installé.

Manuel Technique

À travers cette fonction, nous utilisons principalement trois tableaux. Le premier étant les données de chaque points (ici des coordonnées cartésiennes), les centres de clusters, ainsi qu'un tableau nous donnant le quel de ces centres est le plus proche de chacun des points. Pour modifier les graphiques que nous avons en tant que résultat, il faut modifier quelques données dans le fonction *kmeans*. Les couleurs du graphiques quant à elle sont contrôlées par la fonction *get_ cmap*, qui créer une table de couleurs. C'est grâce à cette table de couleurs, que nous affectons à chaque clusters une couleur différente.

Comme la fonction *KMeans* marche en utilisant la convergence des centres de clusters, il y a un certain ϵ, très petit qui nous permet de vérifier, s'ils convergent ou pas. Pour modifier la valeur de cet ϵ, il faut aller dans la fonction *ite_ movmnt_ cetro(data,centr)* et changer la valeur de *epsi*. La fonction s'arrête également automatiquement, au cas où on rencontrerait un cas de divergence.

4.2.1 Manuel d'Utilisation

Il existe deux façons d'utiliser la fonction KMeans. Tout d'abord, il faut lancer une console *Python*. Une fois cela fait, il faut importer le fichier que nous avons créé en tant que module. Donc ainsi :

```
import kmeans
```

Puis choisir entre deux possibilités :

— Prende des données d'un fichier sur la machine, qui est de forme :

```
223 115
177 122
151 14
128 2
122 213
....
```

Qui représente des coordonnées. Il faut alors utiliser :

```
kmeans.kmeans_file(file_n,k)
```

Où *file_ n* est le chemin vers le fichier de ces données et k le nombre de clusters souhaité.

— La deuxième possibilité est de généré aléatoirement des données. Il faut dans ce cas utiliser :

```
kmeans.kmeans_rand(mini,maxi,n,k)
```

Ici, nous avons *mini*, qui représente la valeur minimale de l'intervalle de génération de données, et *maxi*, qui représente la borne supérieur de cet intervalle. La quantité de données est contrôlé par n, et k le nombre de clusters. La fenêtre graphique affiche également l'ensemble de données avant et après un certain nombres d'itérations, qui sera affiché sur la figure.

Une fois qu'une des deux fonctions a été appelée, nous voyons une figure comme celle-ci :

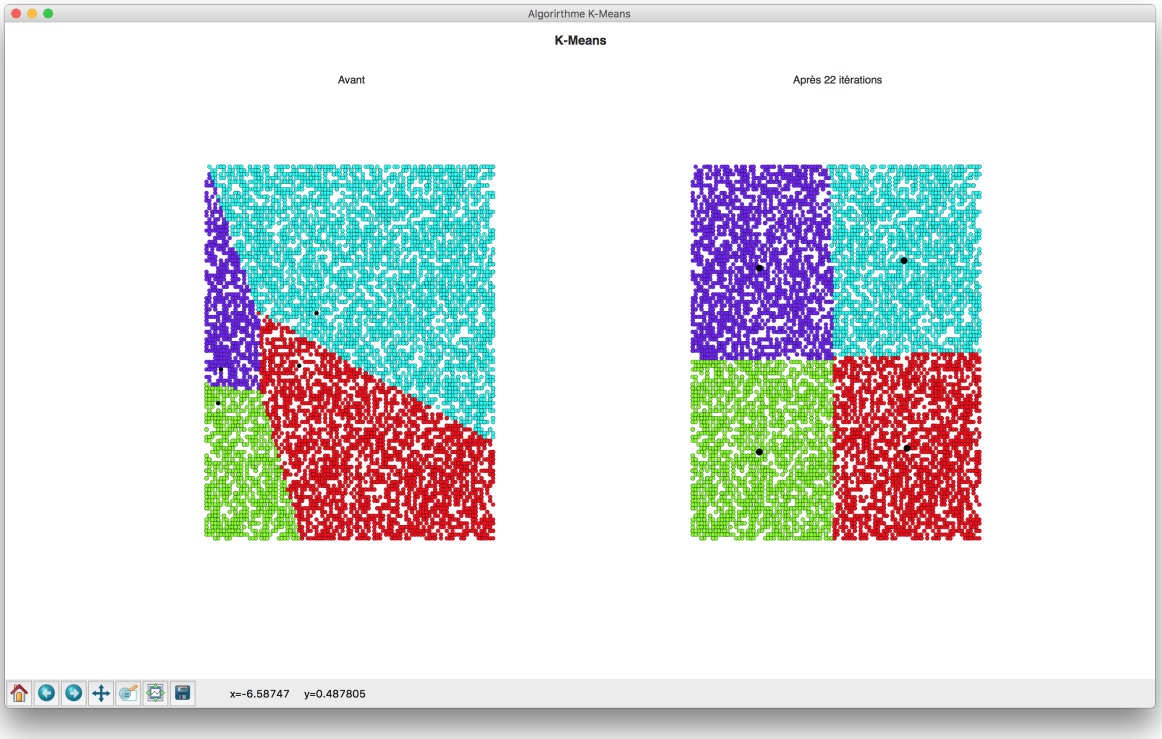

FIGURE 4 – Sortie graphique pour 10000 valeurs entre $[0, 100]$ avec $k = 4$

Et *après* la fermeture de cette fenêtre graphique nous verrons dans la console les coordonnées des différents centres de clusters.

```
>>> kmeans.kmeans_rand(0,100,10000,4)
Coordonnée des centroides :
array([[74, 24],
       [23, 23],
       [73, 74],
       [23, 72]])
```

FIGURE 5 – Sortie console pour 10000 valeurs entre $[0, 100]$ avec $k = 4$

4.3 TF.IDF

Comme PySpark a été fait pour les traitements de données, beaucoup de fonctions qui nous sont utiles dans le cas de TF.IDF, comme dans pleins d'autres cas, sont déjà définies. Cette application lit tous les fichiers ayant l'extension *.txt* et calcul, soit le TF.IDF de tout les différents mots, où que d'un mot en particulier.

Manuel Technique

Pour changer l'extension des fichiers lu, il faut modifier cette ligne de code :

```
1 documents = sc.textFile(dir_path + "/*.txt").map(lambda line: line.
    split(" "))
```

Et modifier la chaîne de caractères *"/*.txt"*.

Manuel d'Utilisation

La toute première chose à faire avant de lancer l'application *tf_idf.py*, est de choisir le répertoire dans le quel l'algorithme va aller prendre les fichiers. Ce chemine est modifiable à la ligne :

```
1 dir_path = "PATH/tf_idf"
```

Une fois que le répertoire a été défini, nous pouvons choisir de ne sortir que le TF.IDF d'un seul mot, dans quel cas nous devons l'entrer ici :

```
1 mot = ""
```

Cependant, si nous souhaitons avoir le TF.IDF de l'ensemble des mots présents dans l'ensemble des documents, il suffit de laisser ce string vide.

```
mot : fonction
tf.idf : 3.044522
```

FIGURE 6 – TF.IDF du mot "fonction"

```
5, 995374: 9.2027, 995717: 3.0445, 1016083: 2.7568, 1020182: 3.45, 1042906: 3.45
, 1046705: 3.45}), SparseVector(1048576, {0: 0.5878, 9685: 3.45, 75075: 3.45, 12
0217: 3.45, 159620: 3.45, 165672: 2.1972, 196957: 3.45, 289190: 3.0445, 332221:
3.45, 350166: 2.5337, 385776: 3.45, 399861: 3.45, 415735: 3.45, 477140: 3.45, 49
2228: 3.0445, 497036: 3.45, 560798: 3.0445, 567961: 1.5517, 596888: 3.0445, 6557
47: 3.3165, 664542: 3.45, 697419: 1.3548, 711207: 0.8109, 753595: 3.0445, 783212
: 1.2528, 788683: 3.45, 848588: 3.45, 854522: 1.5041, 858470: 2.3514, 872673: 0.
9243, 1018566: 3.45, 1043957: 1.5041}), SparseVector(1048576, {0: 0.5878, 90154:
 1.5782, 223888: 3.0445, 227386: 2.7568, 870721: 2.3514, 951255: 2.7568, 969945:
 2.1972, 994775: 2.7568}), SparseVector(1048576, {0: 0.5878, 17870: 4.1274, 3802
0: 1.5041, 50581: 1.0521, 72245: 2.7568, 90154: 1.5782, 91892: 3.4905, 104251: 2
.0637, 150387: 3.0445, 161442: 2.7568, 165672: 2.1972, 170195: 1.5782, 193831: 3
.0445, 199565: 3.45, 214919: 3.45, 227386: 2.7568, 250475: 3.0445, 323369: 3.45,
 348387: 3.45, 385247: 2.3514, 411124: 3.45, 441529: 1.9459, 452525: 1.5782, 502
018: 3.45, 505499: 2.7568, 518463: 3.45, 527939: 6.9, 548650: 3.45, 567961: 3.10
34, 580836: 3.45, 595827: 1.5782, 599682: 3.45, 605228: 6.089, 634804: 3.45, 641
702: 4.7028, 663236: 3.45, 667692: 3.45, 670869: 6.089, 697419: 2.7096, 700713:
3.45, 711203: 1.5517, 711207: 4.0547, 712920: 3.45, 771438: 2.7568, 771492: 2.75
68, 794083: 3.45, 815285: 1.5782, 818294: 3.45, 821646: 3.45, 825872: 1.9459, 82
6638: 5.2357, 828330: 1.5782, 835501: 3.45, 853406: 3.45, 854514: 1.9459, 854522
: 1.5041, 854526: 1.5782, 870721: 2.3514, 872673: 0.9243, 940756: 1.0986, 940760
: 1.6582, 946271: 3.45, 947873: 3.0445, 972130: 3.0445, 992287: 3.0445, 994182:
2.7568, 995374: 3.6811, 997590: 1.9459, 1011643: 3.45, 1018546: 3.45, 1043957: 1
.5041}), SparseVector(1048576, {0: 0.5878})]
```

FIGURE 7 – TF.IDF de tous les mots d'un ensemble de documents

4.4 Notes d'informatique

L'application des notes d'informatique consistait à prendre un fichier donné de format CSV contenant une liste de 50 élèves avec une note en Informatique Théorique ainsi qu'une note en programmation. À ces notes, nous devions appliquer l'algorithme KMeans et l'algorithme de Régression Linéaire. Le fichier correspondant est *notes_info.py*

Manuel Technique

Pour mener à bien les calculs que nous faisons ici nous allons avoir besoin d'utiliser le module *NumPy*. Dans notre cas, le fichier d'entrée est sous la forme :

```
id,Info theorique,Programmation
1,12.5,10
2,13,7.5
3,6,5
```

La première colonne n'a aucune importance pour nous, sachant que ces *id* sont dans l'ordre croissant de 1 à 50. Nous n'utilisons que la deuxième et la troisième colonne ici. Si nous souhaitons utiliser plus de deux colonnes, ou d'autre colonnes que la deuxième et la troisième, il nous suffit de changer les indices sur cette ligne :

```
data = file_in[:,[1,2]]
```

Ici :

```
[1,2]
```

correspond au colonne que nous souhaitons utiliser. Il faut alors effectuer les modifications correspondante dans la suite de l'algorithme

De plus on remarquera que dans le fichier utilisé, nous avons la première ligne qui correspond aux titres de chaque colonnes. La ligne :

```
file_in = np.delete(file_in,0,0)
```

supprime les noms de colonnes NumPy renvoie la valeur de *NaN*, donc qui nous est inutile dans le traitement de données.

Commençons par parler de la partie concernant KMeans puis celle concernant la Régression Linéaire.

Pour affecter des couleurs différentes à chaque cluster, nous utilisons un cycle de couleur, qui est défini ainsi :

```
colors = ['r','b','y','g','c','m']
```

Pour modifier l'ordre du cycle, il nous suffit tout simplement de modifier l'ordre de définition des caractères.
Nous pouvons également modifier la taille de chaque centre de clusters en modifiant la valeur associé à s dans la ligne suivante :

```
plt.scatter(cluster_centers[i][0],cluster_centers[i][1],s=100,c='k')
```

Passons maintenant à la régression linéaire. Dans notre cas, les notes font parties de l'intervalle $[0, 20]$, cependant, il se peut que nous aillons besoin d'un plus grand intervalle. L'intervalle est nécessaire pour tracer la droite de régression. Dans le cas, où nous ayons besoin d'un plus grand intervalle, il nous suffit tout simplement de modifier l'intervalle à cet endroit là :

```
graph(range(0, 20))
```

Manuel d'Utilisation

Pour utiliser l'application des notes il faut tout d'abord aller dans la section *Paramètres* du fichier :

```
################## PARAMETRES ############################
dir_path = PATH
file_name = "exemples_notes.csv"
k=2
##########################################################
```

En effet, la première chose à faire est d'entrer le chemin pointant vers le répertoire contenant le fichier qu'on souhaite étudier dans la variable *dir_path*. Il faut ensuite entrer dans la variable *file_name* le nom complet du fichier. Ici k correspond au nombre de cluster qu'on souhaite avoir.

Maintenant qu'on ai paramétré le fichier, exécutons alors l'algorithme. Pour faire cela, il faut se placer dans le répertoire principale de Spark, ouvrir la console et exécuter la commande suivante :

```
./bin/spark-submit PATH/notes_info.py
```

Tout en remplaçant PATH par le chemin amenant au fichier *notes_info.py*

Résultats

Essayons maintenant l'algorithme et essayons de tirer des conclusions de ces résultats. Si nous lançons avec $k = 2$, nous avons le résultat suivant :

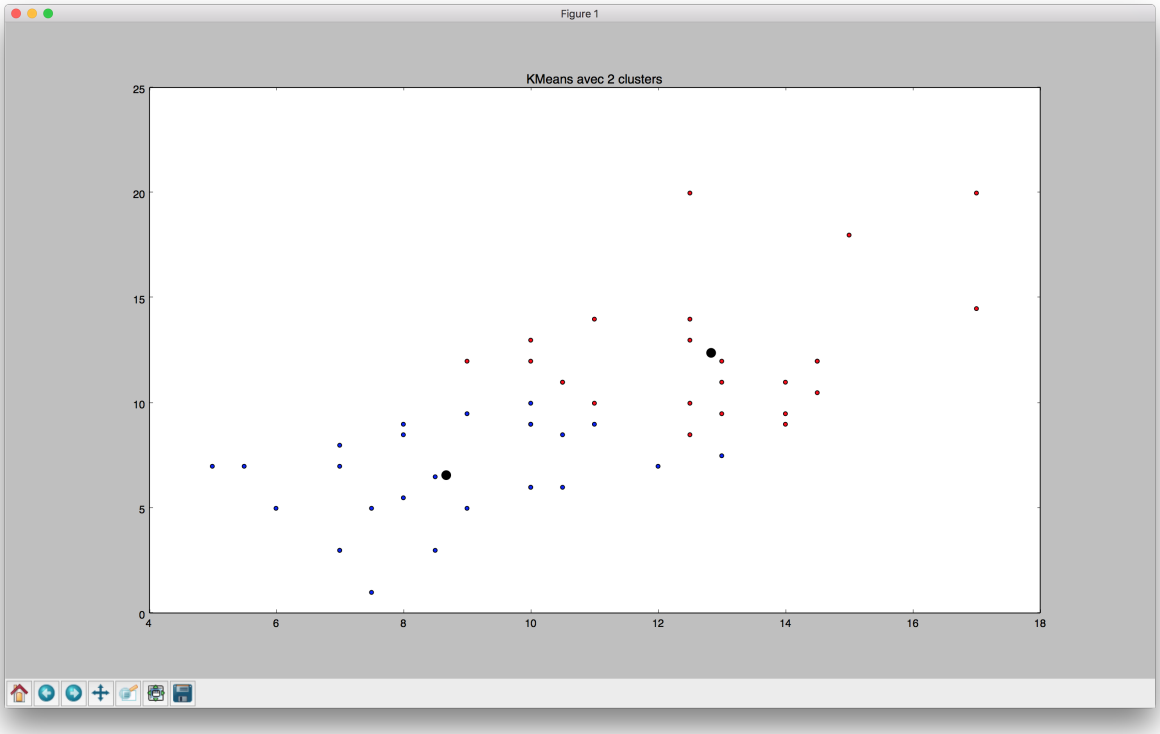

FIGURE 8 – K-Means avec $k = 2$

```
Voici les eleves faisant partie du cluster de centre [ 12.83333333  12.39583333]
[1, 4, 6, 7, 11, 14, 15, 18, 19, 22, 23, 25, 26, 27, 33, 34, 36, 38, 39, 40, 43,
 47, 48, 49]

Voici les eleves faisant partie du cluster de centre [ 8.67307692  6.57692308]
[2, 3, 5, 8, 9, 10, 12, 13, 16, 17, 20, 21, 24, 28, 29, 30, 31, 32, 35, 37, 41,
42, 44, 45, 46, 50]
```

FIGURE 9 – Affichage Console avec $k = 2$

On remarque que cela nous divise le groupe en 2 qui, par déduction, correspondent au bons et au moins bons élèves. On peut également voir quel élève fait parti de quel cluster. En effet, nous affichons le contenu de chaque cluster en affichant également le centre du cluster, permettant de savoir de quel groupe il s'agit. Réessayons, mais avec $k = 4$.

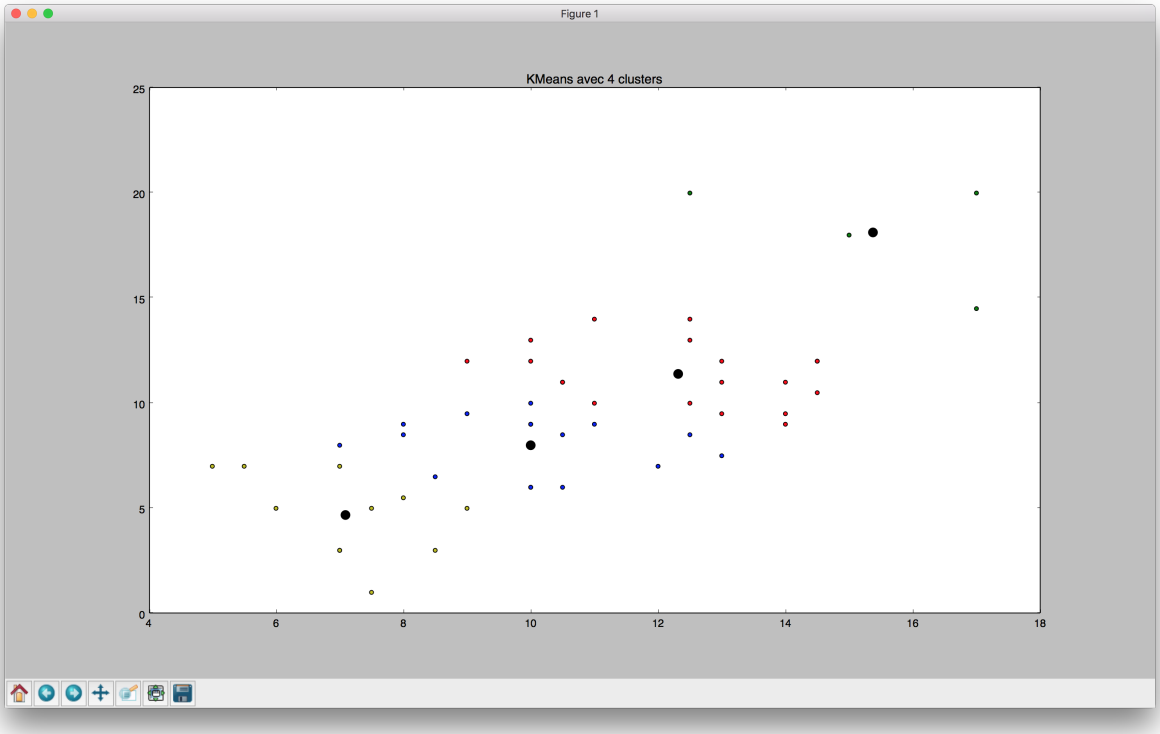

FIGURE 10 – K-Means avec $k = 4$

```
Voici les eleves faisant partie du cluster de centre [ 12.31578947  11.39473684]
[1, 4, 6, 7, 11, 14, 18, 19, 22, 23, 27, 33, 34, 36, 38, 39, 47, 48, 49]

Voici les eleves faisant partie du cluster de centre [ 10.   8.]
[2, 8, 9, 13, 17, 20, 21, 24, 26, 28, 35, 37, 41, 42, 45, 50]

Voici les eleves faisant partie du cluster de centre [ 7.09090909  4.68181818]
[3, 5, 10, 12, 16, 29, 30, 31, 32, 44, 46]

Voici les eleves faisant partie du cluster de centre [ 15.375  18.125]
[15, 25, 40, 43]
```

FIGURE 11 – Affichage Console avec $k = 4$

On peut donc remarquer que KMeans forme des groupes d'élèves en fonction du nombre de cluster qu'on veut, d'où le fait que KMeans est un algorithme de classification. Ici, la visualisation des données est moins importante par rapport au premier exemple que nous avons mis due à la quantité de données.

Parlons maintenant de la régression linéaire. En effet, peu import la valeur de k, la régression linéaire reste inchangée. Voici les résultats, de la régression linéaire, peu importe la valeur de k.

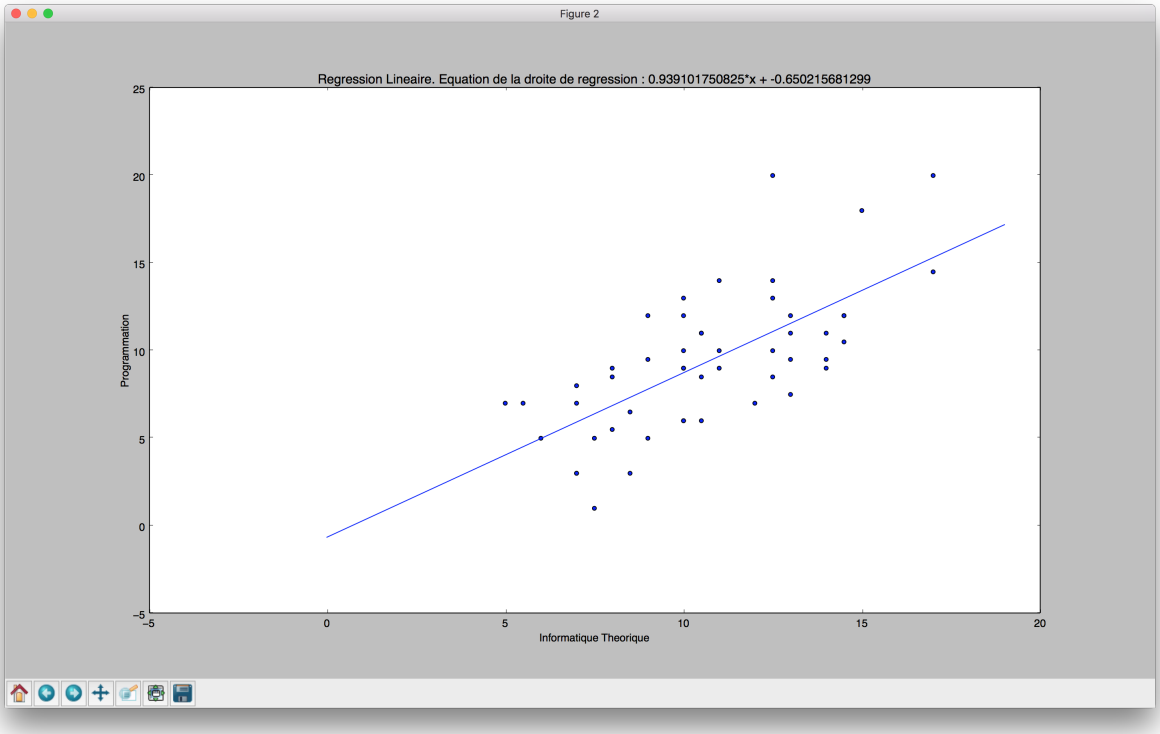

FIGURE 12 – Régression Linéaire

Pour interpréter ce graphique, il suffit de se placer sur une note dans la matière voulu, et remonter vers la droite de régression. Cela nous permet de dire que *si un élève a une notre x en tel matière, alors il devrait avoir y comme note dans l'autre matière.*

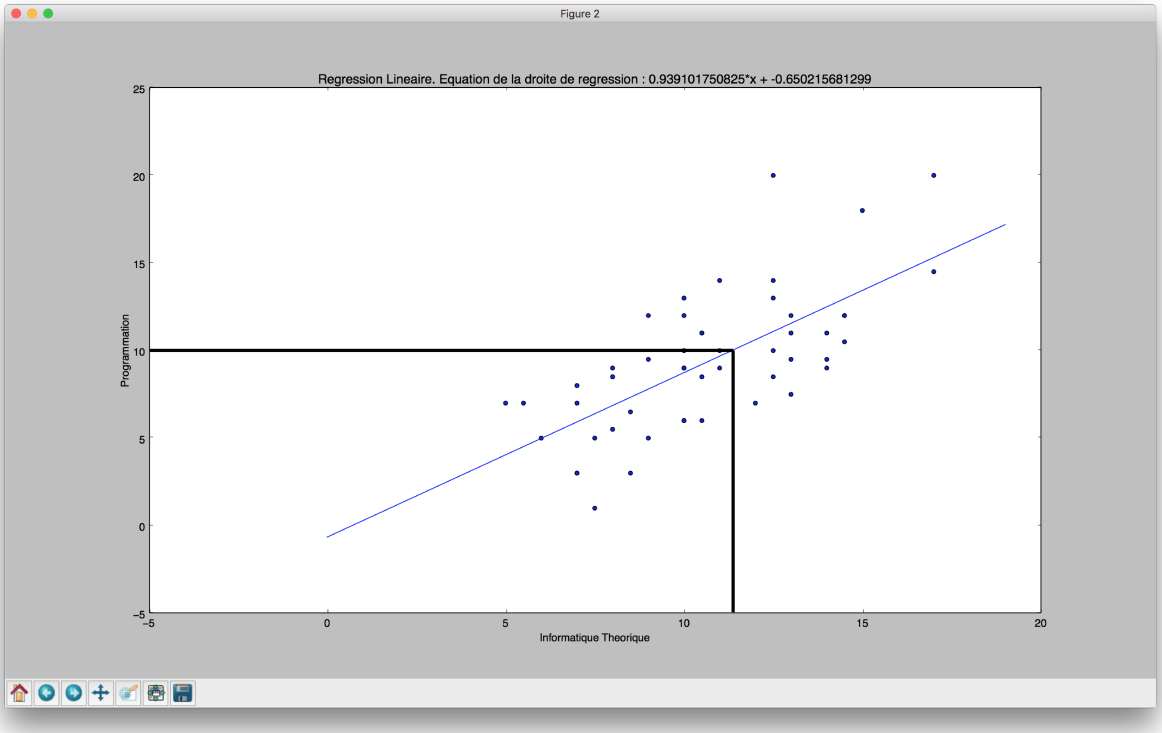

FIGURE 13 – Interprétation Régression Linéaire

5 Conclusion

Pour conclure, le traitement de données représente un enjeu capital dans la société d'aujourd'hui. Le flux d'information émis, partagé, collecté est considérable ainsi tous les secteurs (marketing, télécommunication, surveillance, santé, etc) y voient leur intérêt quant au traitement et à l'analyse de ces informations.

AJOUTER difficultés des algorithmes, car Spark est une technologie avancé par rapport à notre niveau, mais cela a été enrichissant Dans le cadre du projet TIPE, nous avons donc présenté une étude globale de certains algorithmes de traitement de données. Nous avons également exploité ces algorithmes afin qu'ils renvoient des résultats exploitables pour l'utilisateur. Nous avons néanmoins rencontré quelques difficultés au niveau de la programmation algorithmique. En effet nous n'avions que quelques notions du langage de programmation Python avant d'entamer le projet. De plus le framework Spark fut difficile à maîtriser. Bien que les notions abordées dans notre projet et les documentations utilisées dépassent

nos compétences scolaires nous avons tout de même réussi à synthétiser le tout pour présenter un travail cohérent. Ce projet s'est révélé très enrichissant, car en plus de nous avoir entraîné dans un travail de programmation algorithmique, il nous a demandé une longue étude sur le traitement de données. Ce fût une bonne chose que notre sujet ne ce soit précisé qu'au début de l'année de CPI2, car l'étude que nous avons fourni sur le Big Data et ses technologies l'an passé a grandement éclairé notre vision du sujet. Le projet TIPE a donc été bénéfiques que ce soit sur le plan culturel ou scolaire puisque certains d'entre nous restent intéressé par la spécialisation en data science que propose l'EISTI.

Nous remercions le professeur Houcine Senoussi pour son attention dans le suivi de nos recherches ainsi que pour toute l'aide qu'il nous a apportée tout au long du projet.

Bibliographie

1 - https://matplotlib.org/api/pyplot_api.html
2 - Wikipedia
3 - Documentation Spark
4 - Documentation Python et PyPlot
5 - https://fr.wikipedia.org/wiki/R\%C3\%A9gression_lin\%C3\%A9aire
6 - https://fr.wikipedia.org/wiki/K-moyennes
7 - https://fr.wikipedia.org/wiki/TF-IDF
8 - https://fr.wikipedia.org/wiki/Fouille_de_textes
9 - https://en.wikipedia.org/wiki/Text_mining
10 - AstridJourdan_RegressionLin.pdf donné par Houcine Senoussi
11 - https://sites.google.com/site/dataclusteringalgorithms/k-means-clustering-algori
12 - https://en.wikipedia.org/wiki/K-means_clustering
13 - https://www.youtube.com/watch?v=_aWzGGNrcic
14 - https://www.coursera.org/learn/machine-learning/lecture/93VPG/k-means-algorithm
15 - https://fr.coursera.org/learn/text-mining

6 Annexe

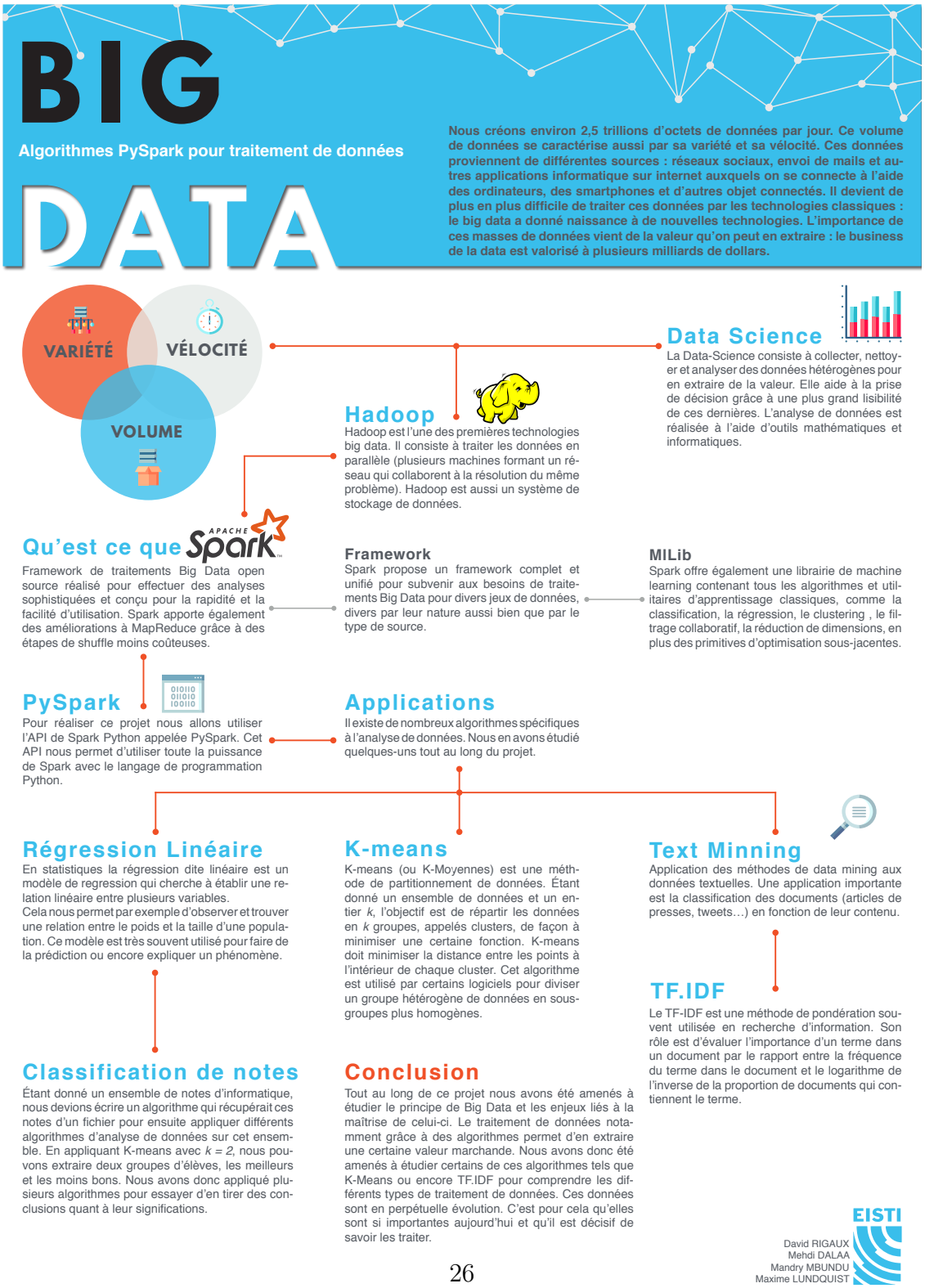

FIGURE 14 – Poster