Healthy-Tomatoes

because rotten tomatoes aren't good for you.

13.12.2017

ELBEZ Samuel 21200353

JACQUETTE Pierrick 21305551

Université Paris Diderot

Table des matières

Remerciements

Nous tenons à remercier Mme Anne-Claire Haury, responsable du cours et M Baptiste Fontaine, enseignant des travaux pratiques pour les nombreux conseils et avis pendant ce semestre.

Vue d'ensemble

Nous vous présentons le rapport du travail effectué dans le cadre du projet de Fouille de Données de master 2. Le projet est la réalisation d'un système de prédictions de la réussite d'un film basé sur son contenu. Le coeur du système est réalisé avec l'algorithme de TF IDF. L'implémentation de l'algorithme semblait assez simple, le module SkLearn vu en tp, nous permet d'avoir des algorithmes dans le domaine du machine learning.

Dans la première partie du rapport nous allons vous présenter l'idée générale du projet, ses objectifs et la méthodologie que l'on a mis en oeuvre. Dans une seconde ce que nous avons mis en oeuvre puis les résultats obtenues. Après nous aborderons les difficultés rencontrées pour enfin parler des extensions possibles.

Les technologies utilisées sont python 3.5 et elasticsearch. Les données des films dont on a besoin pour ce projet viennent du site Kaggle : https://www.kaggle.com/tmdb/tmdb-movie-metadata/data. Les différentes libraires pour ce projet sont sklearn, scipy, numpy et elasticsearch.

Les sources sont sur github, une documentation est présente sur une autre branche.g

Objectif

Le projet vise à prédire avant de tourner un film si il va être un succès ou un échec.

Théorie et méthodologie

Tout d'abord il fallait pour chaque film savoir si il était un succès ou échec. On disposait d'un champ "vote_average" variant entre 0 et 10. On a donc calculé la moyenne, médiane et écart-type de ce champs pour tous les documents dont on disposait, même sur l'ensemble de test afin de savoir la note limite entre un succés et un échec. Cela nous a permis lors de l'insertion dans la BDD de rajouter un champ SUCCESS valant 0 ou 1.

Les méthodes d'apprentissage supervisées visent à apprendre et construire un modèle à partir de grosses volumes de données. La machine doit être capable de classifier afin de prédire des nouvelles données. Les premières questions que l'on a du se poser, c'était quelles sont les données dont on dispose ? En quoi elles pourraient-nous être utile ? On avait 4800 documents à notre disposition avec chacun plein de champs dont on ne voyait pas l'intérêt.

Les données étant pas numérique on a du les rendre interprétable pour la machine. On s'est donc demandé comment transformer en objets numériques sur lesquels il serait possible de travailler ? Beaucoup de champs qui nous intéressaient étaient du texte, on a donc décidé de mettre en oeuvre la technique TF-IDF. Chaque dimension du vecteur correspond alors au vocabulaire de l'ensemble des documents. Les mots importants sont mis en valeur. L'avantage que présente ce vecteur est qu'il permet de faire des calculs de similarité et de comparer des documents différents entre eux.

Par la suite nous nous sommes demandés, si l'on ne pouvait pas faire mieux, en réfléchissant l'on a voulu testé l'algorithme des k plus proches voisins puis comme on adorait les statistiques on a fait des tests sur l'algorithme des arbres de décision et des randoms forest.

Technique et implémentation

Nous allons expliquer comment nous stockons nos données et les méthodes de méthode learning que l'on utilisent.

Pour la base de données, on a opté pour elasticSearch. Pourquoi ce choix ? Elastic propose un moteur de recherche et d'analyse distribuée en temps réel. Cela nous permet d'explorer et de voir les données à grande échelle. On trouvait cela parfaitement adapté à l'idée de datamining.

On a pensé dés l'insertion dans la base et donc pendant le parsage des fichiers à créer deux tables : un ensemble d'entrainement et un ensemble de test.

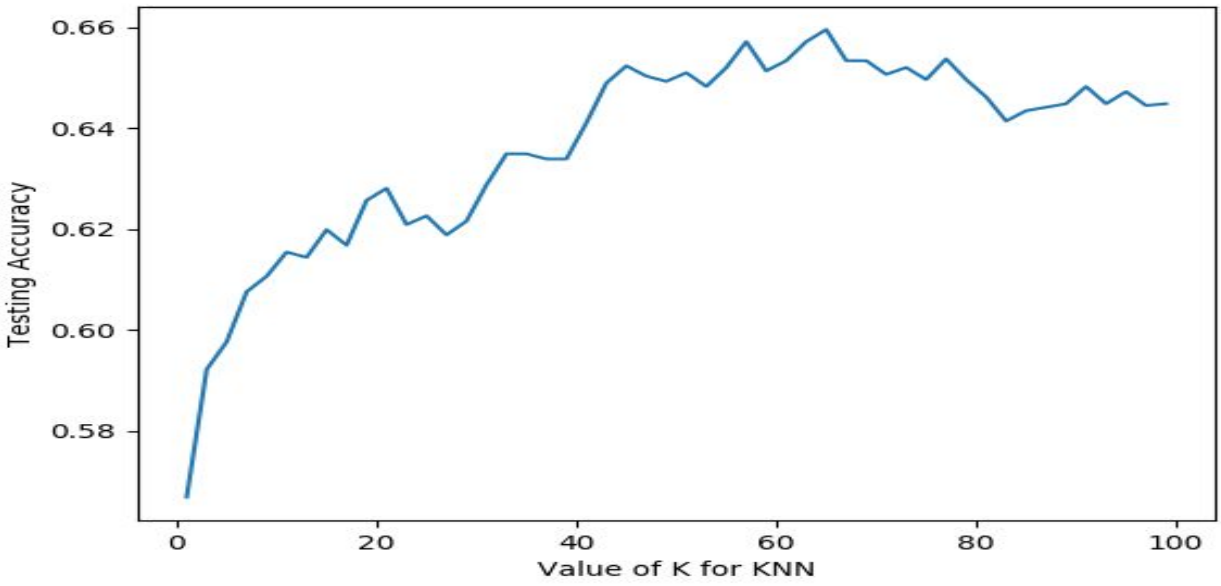
On se sert du résultat de l'algorithme de TF-IDF afin d'appliquer un algorithme naïf, on regarde tous les champs indépendamment. Pour chaque film de l'ensemble de test, on regarde si chaque mot apparaît plus comme un succès ou un échec, on incrémente un compteur si c'est un succès, on le décrémente si c'est un échec et ne le prend pas en compte si le mot n'existe pas.

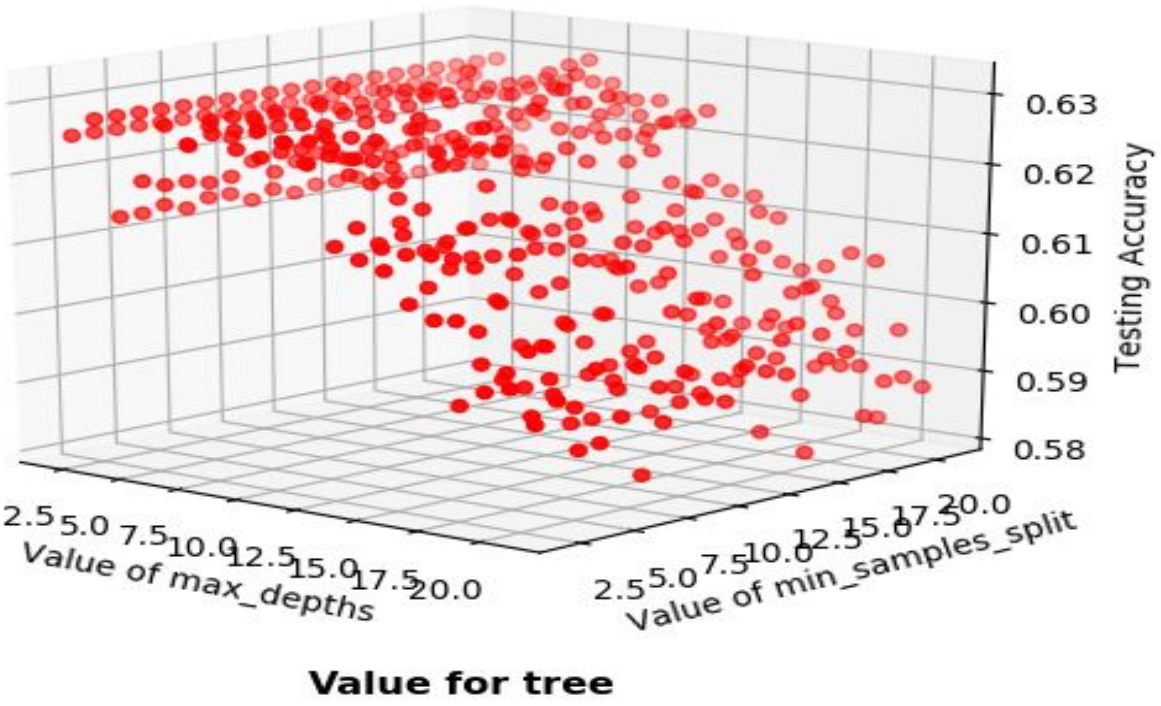
On peut dans ce cas de figure calculer le nombre de mot qui sont perdus (ceux qui n'apparaissent pas dans l'ensemble d'entraînement) pour chaque film. Cela nous permet de savoir de combien notre résultat est précis.

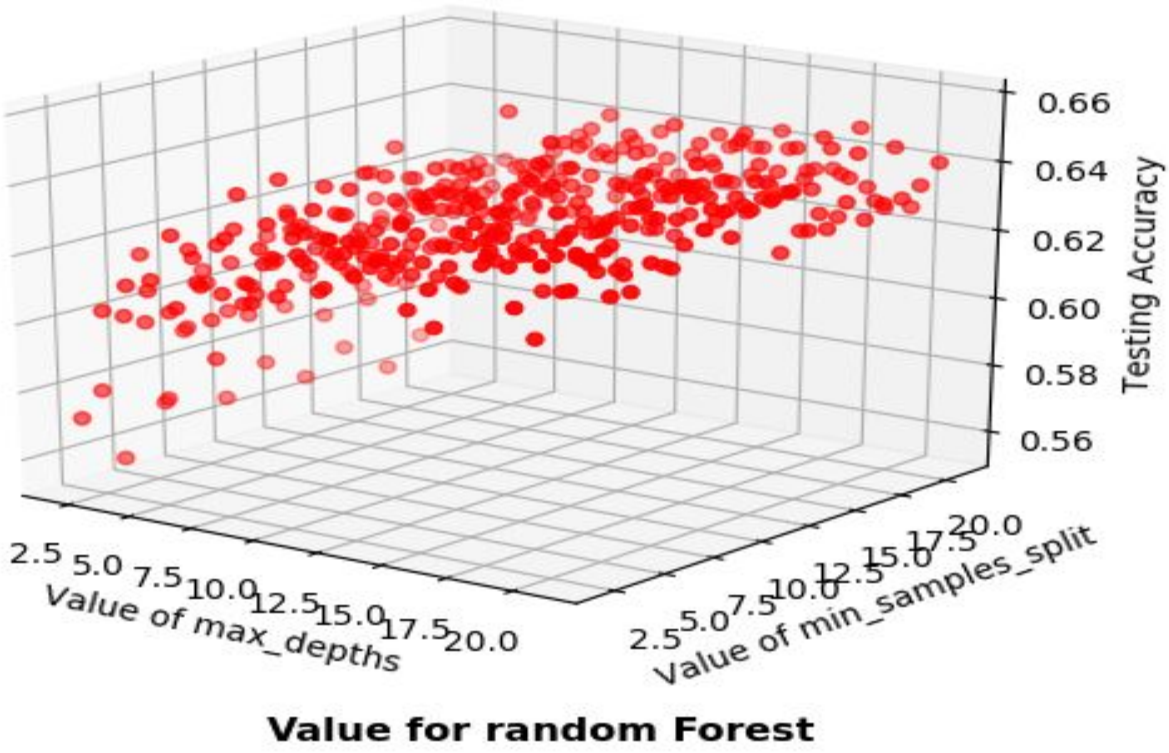
Nous avions une représentation des champs importants de manière interprétable pour la machine, il restait plus qu'à ..... Et là ce fut la question, quel algorithme choisir ! On avait en tête celui des plus proches voisins. Mais il fallait un k, après quelques secondes de réflexion, on a pensé à la technique de validation croisé. Cela permet d'obtenir le meilleur score possible et connaître le k correspondant. Ce que l'on a implémenté par la suite afin d'obtenir le k idéal et pouvoir l'appliquer à notre ensemble de test.

Quand on aime on continu, on a donc regardé les valeurs que l'on obtient avec l'algorithme des arbres de décision et des forêts aléatoires.

Graphiques pour le choix des paramètres des algos







Les résultats

Pour les méthodes suivantes, notre accuracy est :

- naive Bayes : 65%
- k plus proches voisins : 64%
- arbre de décision : 62%
- random forest : 62%

Les difficultés

La première difficulté fut de comprendre le projet, essayer de le structurer. Le sujet du projet étant tellement large, par où commencer ? On a donc décidé d'avancer par étape, d'abord récupérer les données, puis les stocker, les récupérer de la base de données… On avait pas de vision globale mais une fois chaque étape finie, on arrivait à connaître la suivante. Le projet a donc avancé par itération locale en reliant à chaque étape avec la précédente.

La seconde fût elasticSearch, il ne fait pas exactement ce que l'on voulait, par exemple quand on cherché tous les films d'un genre, il nous disait qu'il y en avait 499 et nous renvoyé uniquement une liste de taille 10. On mis du temps avant de comprendre d'où venait le problème. Par ailleurs elasticSearch renvoie des documents, on se retrouvait avec un objet renvoyé par la BDD assez complexe par son imbrication….

Le nom et le prénom des personnalités sont séparés par des espaces et donc traités indépendamment par la tf-idf.

Les extensions possibles

Considérés tous les champs fourni en appliquant les bons poids pour voir si cela permet d'être précis dans nos prédictions.

Ne pas faire un tf-idf sur les genres mais appliquer un vecteur dessus.

Une interface graphique ou textuelle permettant à l'utilisateur de rentrer un nouveau film avec tous les champs nécessaire.

Conclusion

Pour conclure, nous voulions rappeler que ce projet a été enrichissant pour nous. On comprend mieux comment certains outils de la vie quotidienne fonctionnent. On est capable de prévoir pour pouvoir décider en fonction de l'expérience.