





Université Paris Diderot

## Table des matières

|  |          |
|--|----------|
| <b>Table des matières</b>                                | <b>1</b> |
| <b>Remerciements</b>                                     | <b>1</b> |
| <b>Vue d'ensemble</b>                                    | <b>2</b> |
| <b>Objectif</b>  | <b>3</b> |
| <b>Théorie et méthodologie</b>                           | <b>3</b> |
| <b>Technique et implémentation</b>                       | <b>4</b> |
| <b>Graphiques pour le choix des paramètres des algos</b> | <b>5</b> |
| <b>Les résultats</b>                                     | <b>7</b> |
| <b>Les résultats</b>                                     | <b>7</b> |
| <b>Les difficultés</b>                                   | <b>8</b> |
| <b>Les extensions possibles</b>                          | <b>8</b> |
| <b>Conclusion</b>  | <b>8</b> |

## 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 prédiction 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 de datamining.

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 librairies pour ce projet sont sklearn, scipy, numpy et elasticsearch.

Dans les deux fichiers csv contenant les données, on a commencé par distinguer les champs dont on avait besoin et que l'on allait utiliser. Le genre, les keywords, les compagnies de production, les 6 premiers acteurs et le réalisateur nous serviront pour la tf-idf. La colonne "vote\_average" nous permettra de savoir si c'est un succès ou un échec. Ce sont les champs qui sont plus susceptibles d'attirer un public pour voir un nouveau film. Le champ "budget" a été ignoré car on peut avoir un très bon film avec un petit budget et inversement. Le titre est également ignoré, car on a voulu ignorer le concept de suite ce qui le seul cas intéressant pour ce champ.

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

## Objectif

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

## Théorie et méthodologie

Tout d'abord, il fallait pour chaque film savoir s'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 champ pour tous les documents dont on disposait, même sur l'ensemble de tests afin de savoir la note limite entre un succès et un échec, ici, on a pris la médiane qui était à 6.2. Cela nous a permis lors de l'insertion dans la BDD de rajouter un champ SUCCESS valant 0 ou 1.

Au début du développement, nous avons écrit sur papier le déroulement de notre algorithme de base. Le stockage dans une tf-idf puis l'utilisation des données de celle-ci. Pour obtenir un résultat, nous avons écrit notre propre version de Naive Bayes. Cette version est beaucoup plus lente que Bernoulli NB de sklearn et moins précise, mais elle nous donne néanmoins des informations intéressantes telles que le nombre de champs qui n'apparaît pas dans notre tf-idf de l'ensemble de test. Pour des raisons de performances, les lignes d'appel dans le fichier "main.py" ont été placées en commentaires.

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

Les données choisies n'étant pas numérique, on a dû les rendre interprétables pour la machine. On s'est donc demandé comment transformer en objets numériques sur lesquels il serait possible de travailler ?

Nous avons 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 demandé, si l'on ne pouvait pas faire mieux, en réfléchissant l'on a voulu tester 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 datamining que l'on utilise.

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 dès l'insertion dans la base et donc pendant le passage des fichiers à créer deux tables : un ensemble d'entraînement et un ensemble de tests.

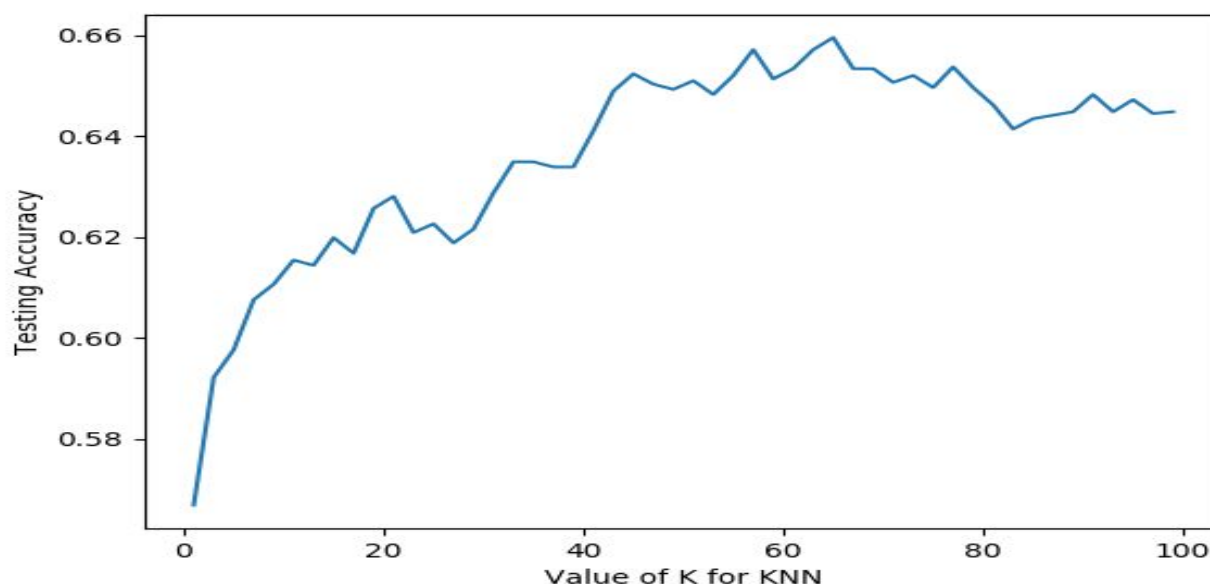
On se sert du résultat de l'algorithme de TF-IDF afin d'appliquer un algorithme naïf, on regarde tous les champs indépendamment. Pour chaque film de l'ensemble de tests, on regarde si chaque mot apparaît plus comme un succès ou un échec, on incrémente un compteur si c'est un succès, on le décrémente 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 mots 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.

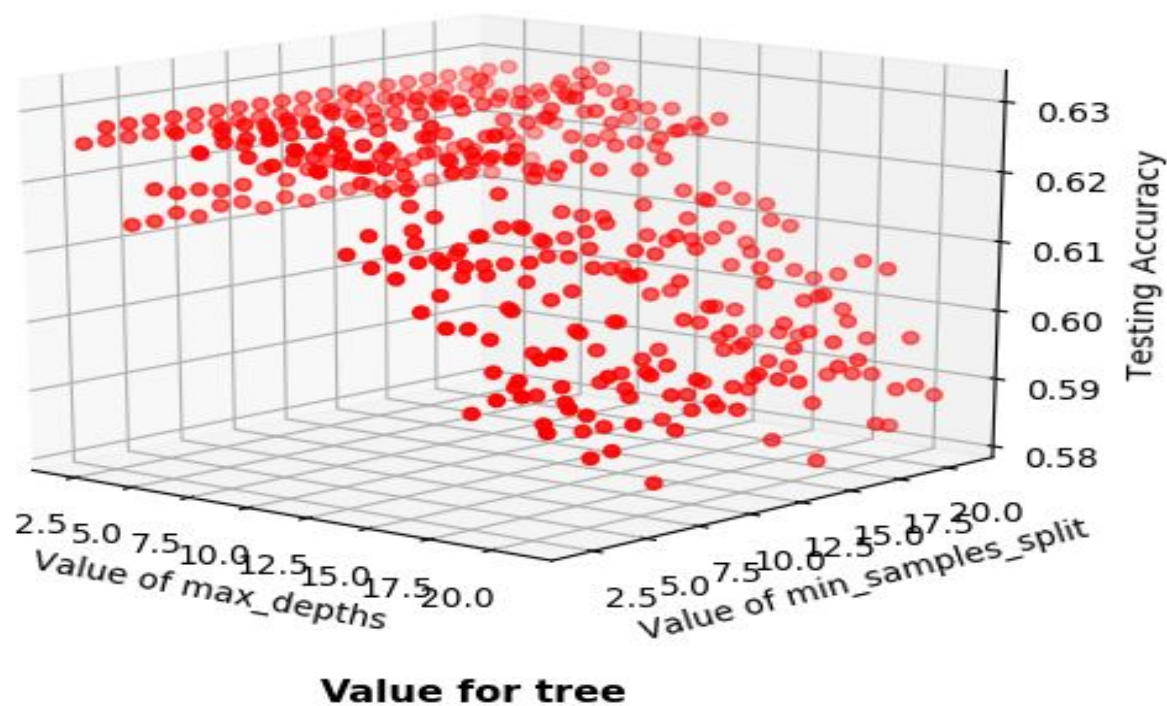
Nous avons une représentation des champs importants de manière interprétable pour la machine, il restait plus qu'à ..... Et là ce fut la question, quel algorithme choisir ! On avait en tête celui des plus proches voisins. Mais il fallait un  $k$ , après quelques secondes de réflexion, on a pensé à la technique de validation croisée. 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 tests.

On a parallélisé les algorithmes de  $K$  voisins et random voisin. 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.

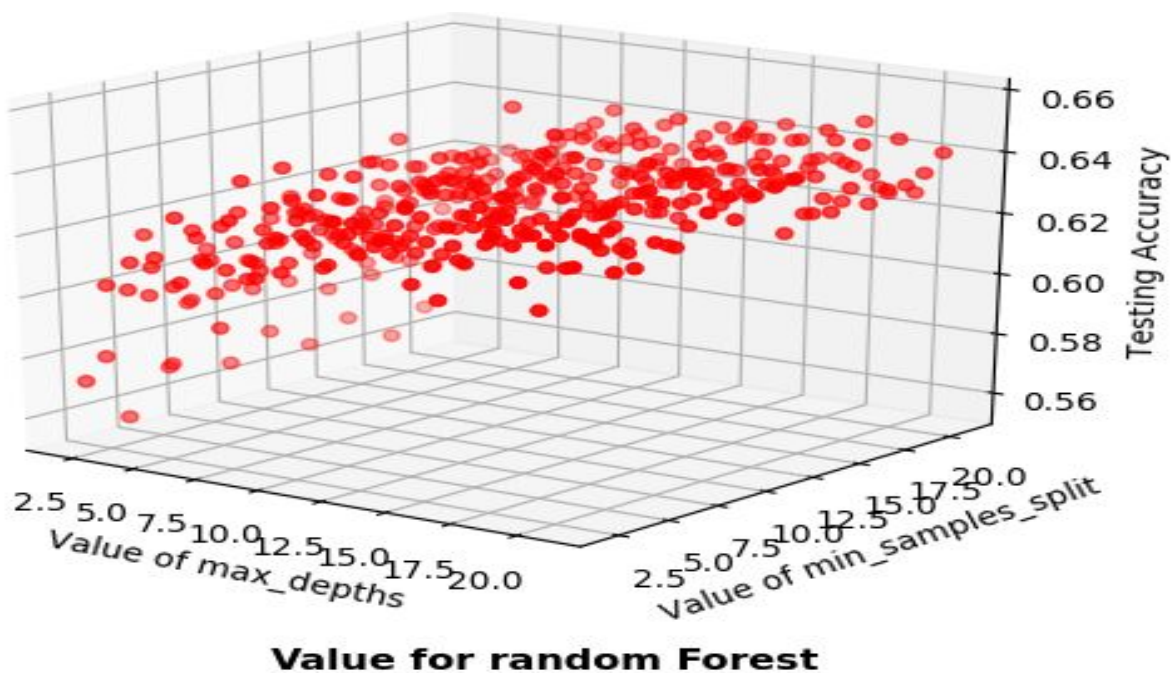
## Graphiques pour le choix des paramètres des algos



On a testé pour différentes valeurs de  $k$  afin de connaître le nombre de voisins optimal.



On a fait la même chose pour les arbres pour la profondeur et combien de noeuds internes.



On a fait la même chose pour les arbres pour la profondeur et combien de noeuds internes.

## Les résultats

Pour les méthodes suivantes, notre accuracy est :

- my naive Bayes : 63%
- naive Bayes : 65% (precision 65 % et recall 64%)
- k plus proches voisins : 61% (precision 62% et recall 61%)
- arbre de décision : 63% (precision 63% et recall 63%)
- random forest : 66% (precision 66% et recall 66%)

Après avoir exécuté plusieurs algorithmes, nous obtenons des résultats aux alentours de 61~65%. Ces résultats sont corrects sans pour autant être fantastique. On peut émettre les hypothèses suivantes :

- les données ne sont pas en nombre suffisant ce qui produit une perte dans le détail
- on ne prend pas en compte la courbe des tendances, par exemple dans les années 90, le genre action est saturé et cela peut fausser les résultats des films à venir
- des champs que nous avons ignorés auraient pu être utilisés

## Les résultats

- my naive Bayes : 250s
- naive Bayes : 1s
- k plus proches voisins : 46s
- arbre de décision : 147s
- random forest : 65s



## 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 étapes, d'abord récupérer les données, puis les stocker, les récupérer de la base de données... Nous n'avions 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 fut elasticSearch, il ne fait pas exactement ce que l'on voulait, par exemple quand on cherchait 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 mit 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...

## Les extensions possibles

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

Appliquer un poids à certains champs pour influencer les résultats, cela n'a pas été fait car il est difficile de déterminer qui est quoi, certains acteurs sont parfois des réalisateurs.

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.