Projet Innovation P5S400 - Jalon 3

**L’IA au service des Ressources Humaines**

FEILLET Eva - HAIGNERE Carla - MOREAU PERNET Baptiste - OMARI BETAHI Yassine

Table des matières (à faire avec Word)

**Introduction**

**HR Toolbox - Présentation**

**Outils théoriques et pratiques**

**Application : démarche au cas par cas**

**API et passage à l’échelle**

**Conclusion**

Explications sur le plan

Introduction: Rappel du contexte (Pôle Emploi, idée initiale du projet) + Changement de projet (raisons) => Nouveau projet

1. idée générale HR Toolbox
2. nos données + outils de travail + méthode de travail (cycle de travail data science )
3. les modules au cas par cas (but, données, preprocessing, training, testing), difficultés rencontrées sur les datasets / l’implémentation de certains modules
4. API et industrialisation

Conclusion générale sur le projet (prise de recul, ce que nous avons bien fait, ce que nous aurions pu faire autrement, ce que nous avons appris, manière de prolonger le projet)

**Introduction**

Notre projet, l’IA au service des ressources humaines, a du évoluer au cours des événements qui jalonnent son histoire. Initialement, en première année, nous étions partis d’une idée proposée par le Lab RH. Après avoir cadré le projet et décidé de construire une plateforme de recommandation d’expériences apprenantes destinées aux actifs et aux chercheurs d’emplois, nous avions eu besoin de données de meilleure qualité et en plus grand nombre pour nourrir l’apprentissage de nos algorithmes. C’est là qu’intervient le partenariat que nous avons mis en place avec Pôle Emploi.

Cette coopération nous a permis d’accéder à un grand volume de CVs anonymisés via une plateforme dédiée, Teralab. Cependant, force est de constater à quel point ces outils étaient insuffisants. Les données, certes volumineuses, étaient peu dense (*sparse data*), des problèmes d’encodages étaient visibles, de nombreux champs étaient obscurs ou inutilisables. Sans compter les contraintes techniques et pratiques posées par la plateforme: problèmes récurrents d’authentification, caractères spéciaux essentiels à l’écriture du code désactivés, bugs récurrents sur l’interface… Malgré notre volonté et notre persévérance, le projet s’enlisait. Il était donc temps de prendre un décision: continuer sur une route incertaine, ou ouvrir une nouvelle voie.

Nous avons donc décidé de continuer notre projet en transformant notre outil afin de pouvoir s’affranchir de Teralab. Comme, pour des raisons de confidentialité, les données de Pôle Emploi ne pouvaient être extraites de la plateforme, nous avons cherché de nouvelles données en open source. A partir de ces nouvelles données, nous avons eu l’idée d’une boîte à outils pour les ressources humaines, en gardant à l’esprit notre souhait initial: construire un outil permettant d’améliorer le bien-être au travail et l’employabilité de nos utilisateurs. HR Toolbox était né.

**HR Toolbox - Présentation**

L’outil que nous avons développé a de multiples facettes. Il est composé de plusieurs modules fournissant des recommandations à partir de l’analyse d’indicateurs RH. Le but de notre outil est d’améliorer le bien-être et l’employabilité des employés et de faciliter le recrutement. Nous gardons l’âme de notre premier outil en intégrant les référentiels de compétences, de métier et de ressenti que nous avions élaboré précédemment, et surtout en fournissant une fonctionnalité de recommandation d’expériences apprenantes. La différence majeure est que nos données concernent essentiellement des profils techniques orientés informatique. Nous avons donc construits nos modèles à partir d’une population bien particulière. Cependant, la méthodologie que nous avons utilisée s’étend bien évidemment à tout autre domaine d’activité, donc notre travail peut être prolongé dans un cadre plus général.

HR Toolbox se compose de 5 fonctionnalités principales, ou modules.

1. Turnover prediction
2. Happiness at work
3. Recrutement
4. Expériences apprenantes
5. Facteurs de succès

Ces fonctionnalités sont basées sur des modèles issus du machine learning et de méthodes de matching. *Toutes n’ont pas été développées jusqu’au même degré de maturité, étant donné la richesse relative des jeux de données sur lesquels elles s’appuient. finir sur qqch de plus positif*

**Outils théoriques et pratiques**

*Dataiku*

Nous avons utilisé le logiciel de data science Dataiku DSS au cours de notre projet. C’est une plateforme collaborative qui permet d’explorer des jeux de données, de faire du prototypage, construire des modèles et les partager de manière efficace. Nous avons codé en Python dans des notebooks Jupyter et nous avons aussi utilisé l’interface graphique de Dataiku.

Exemple d’environnement de travail:

*Machine Learning*

Nous nous sommes principalement reposé sur notre expérience accumulée en recherchant des bonnes pratiques et des exemples d’implémentations sur des forums comme Stack Overflow, ainsi que sur le cours de Fragkiskos Malliaros *Introduction to Machine Learning* dispensé cette année à Centrale.

Nous avons fait appel à des techniques d’apprentissage supervisé (régression, classification) et non supervisé (clustering).

*Démarche : Cycle de travail du data scientist*

Le travail d’un data scientist suit un processus bien particulier. Celui-ci est itératif et peut être schématisé selon le cycle suivant:

**Application : démarche au cas par cas**

**=> NB: on suit la même méthode que pour les projets de ML titanic / final report project**

*Données*

Nous avons utilisé trois jeux de données open source proposés sur la plateforme de data science Kaggle. Nous nous y référerons par les noms suivant:

* Human Resources / *dataset n°1*
* Google Jobs / *dataset n°2*
* Kaggle Survey / *dataset n°3*

Le premier comporte des indicateurs issu du domaine des ressources humaines sur les employés d’une entreprise, du nombre d’heures passées par mois au bureau à la note obtenue au dernier entretien RH en passant par le niveau de satisfaction global.

Le deuxième a été obtenu par scrapping d’un site d’annonces de postes à pourvoir et contient uniquement des offres postées par Google.

Le dernier est issu d’un sondage réalisé par Kaggle auprès de ses utilisateurs en 201? et renseigne entre autre sur les média d’apprentissage et le parcours académique de ceux-ci.

Ces données se concentrent donc sur un type très précis de profil : des personnes avec un profil plutôt technique tendance informatique voire très pointu en data science. Cependant, comme expliqué plus haut, la méthodologie que nous avons utilisée s’étend bien évidemment à tout autre domaine d’activité.

Liens vers la source des datasets:

[1]<https://www.kaggle.com/colara/human-resource>

[2]<https://www.kaggle.com/niyamatalmass/google-job-skills/home>

[3] <https://www.kaggle.com/rayjohnsoncomedy/job-skills/data>

*Démarche*

Pour les fonctionnalités requérant du machine learning, voici la démarche générale que nous avons suivie, de l’exploration des données à l’évaluation du modèle.

1. Exploration des données
2. Cleaning / preprocessing
3. Feature engineering
4. Training
5. Validation / choix de l’algo
6. Testing, performances finales

Dans les explications qui suivent, nous décrivons aussi les fonctionnalités n’exigeant pas de machine learning, mais utilisant plutôt des algorithmes de matching.

A.Turnover prediction

|  |
| --- |
| **But**   * Prédire si un employé est susceptible de quitter l’entreprise à court terme, ce qui correspond à une tâche classification binaire en machine learning. |

Workflow

1. Exploration des données

Nous utilisons le premier jeu de données: “Human Resources”. Les données brutes sont contenues dans un fichier csv de 15 000 lignes fois 10 colonnes. Les champs renseignés ou features sont les suivantes:

satisfaction\_level : niveau de satisfaction de l’employé

last\_evaluation: note obtenue au dernier entretien RH

number\_project: nombre de projets auxquels participe l’employé

average\_montly\_hours: nombre moyen d’heures travaillées

time\_spend\_company: nombre d’années d’ancienneté

Work\_accident: si l’employé a déjà eu un accident du travail (0 ou 1)

left: si l’employé va ou a quitté l’entreprise (0, 1)

promotion\_last\_5years: si l’employé a été promu au cours des 5 dernières années (0, 1)

category: département dans lequel l’employé travaille

salary: catégorie de rémunération

Interface graphique sur Dataiku: dataset brut

Etude statistique

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **left** | **promotion\_last\_5years** | **sales** | **salary** |
| **count** | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999 | 14999 |
| **unique** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 10 | 3 |
| **top** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | sales | low |
| **freq** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 4140 | 7316 |
| **mean** | 0.612834 | 0.716102 | 3.803054 | 201.050337 | 3.498233 | 0.144610 | 0.238083 | 0.021268 | NaN | NaN |
| **std** | 0.248631 | 0.171169 | 1.232592 | 49.943099 | 1.460136 | 0.351719 | 0.425924 | 0.144281 | NaN | NaN |
| **min** | 0.090000 | 0.360000 | 2.000000 | 96.000000 | 2.000000 | 0.000000 | 0.000000 | 0.000000 | NaN | NaN |
| **25%** | 0.440000 | 0.560000 | 3.000000 | 156.000000 | 3.000000 | 0.000000 | 0.000000 | 0.000000 | NaN | NaN |
| **50%** | 0.640000 | 0.720000 | 4.000000 | 200.000000 | 3.000000 | 0.000000 | 0.000000 | 0.000000 | NaN | NaN |
| **75%** | 0.820000 | 0.870000 | 5.000000 | 245.000000 | 4.000000 | 0.000000 | 0.000000 | 0.000000 | NaN | NaN |
| **max** | 1.000000 | 1.000000 | 7.000000 | 310.000000 | 10.000000 | 1.000000 | 1.000000 | 1.000000 | NaN | NaN |

On remarque que beaucoup de champs sont renseignés par ‘NaN’ id est not a number. Nous allons résoudre ce problème par la suite en nettoyant le jeu de données. On remplacera les données ‘NaN’ par des informations moyennées ou on supprimera les lignes correspondantes, et on transformera les catégories ordinales en catégories cardinales.

*Commentaire global à traduire, je l’avais écrit en anglais dans les notebooks sur dataiku*

On average:

- employees have a satisfaction rate of 0.61, not bad but quite low if you expect people to be happy at work;

 they had good grades at their last evaluation (0.72);

 they work on more than 3 projects;

 they spend 7 hours per day at the office;

 they have been in the company for 3 years and a half;

- work accidents are rare;

- few employees got a promotion in the last 5 years: 2% of the staff, this is very low!

20% of the employees want to leave the company

Our goal:

- explain why they want to leave,

- find which categories of employees want to leave

- train a model able to predict who will leave the company

- evaluate the performances of this model

Conclusion:

the salary does not have an impact on the turnover

people who leave either have very few projects or a lot of projects

people tend to leave the company after 3 years at the company

employees with a higher salary are less likely to leave the company (?)

Interpretation:

a promotion is a reward that too few employees were given, so they are less motivated

people with too few projects want to find something more interesting elsewhere

people with too many projects and who spend too much time at work wish to have a more balanced lifestyle

salaries (?)

1. Cleaning / preprocessing

# Convert Department into a numeric feature

# Step1: rank the departments according to their turnover rate

df[['department', 'left']].groupby(['department'], as\_index=False).mean().sort\_values(by='left', ascending=False)

# Step2: transposition on the basis of the ranking

df=df.replace(['hr', 'accounting', 'technical', 'support', 'sales', 'marketing', 'IT', 'product\_mng', 'RandD', 'management'], [0,1,2,3,4,5,6,7,8,9])

# Convert salary into numerics

# Step1: ranking

df[['salary', 'left']].groupby(['salary'], as\_index=False).mean().sort\_values(by='left', ascending=False)

#Step2: transposition

df=df.replace(['low', 'medium', 'high'], [0,1,2])

1. Feature engineering

Nouvelles *features* :

Comment l’employé se sent-il?

- investment (or workload) : average\_monthly\_hours \* number\_project

- reward : promotion + salary (+last\_evaluation?)

- sanity : investment, satisfaction\_level, work\_accident (+reward) => voir selon les services d’activité)

Est ce que l’employé est un membre précieux pour la communauté?

- efficiency : number\_project / average\_monthly\_hours

- experience : number of projects \* time spent in the company

1. Training

Nus avons testé plusieurs algorithmes sur trois sets de données différents, suivant le nombre de colonnes conservées. Les algorithmes les plus performants dans notre cas utilisaient des arbres de classification.

Ici, sur les données brutes, le meilleur score au sens de l’Accuracy (= pourcentage de prédictions justes) est obtenu avec un ExtraTreesClassifier et vaut 99.0%.

1. Validation / choix de l’algo

Avec la prédiction via Dataiku, le meilleur algorithme est XGBoost. L’accuracy vaut aussi 98.6%, ce qui est légèrement inférieur à la performance espérée au vu de nos premiers effets peu optimisés. Cependant, nous sommes aussi tributaires du split des données entre training set et testing set, qui n’est pas le même pour les deux entraînements, donc les scores ne sont pas vraiment comparables.

1. Testing, performances finales

Notre tâche de ML est une classification binaire. La matrice de confusion obtenue avec le XGBoost est la suivante:

Tableau récapitulatif des indicateurs de performance:

Courbe ROC

L’importance relative des *features* pour prédire leur départ ou non de l’entreprise est résumée ci-dessous:

On remarque que dans le cas particulier de cette entreprise, la colonne la plus pertinente est *efficiency*  que nous avions créée à partir de number\_project et average\_monthly\_hours. Surprenament, les *features* associées à la récompense ont un faible poids relatif dans la prédiction. Il semble donc que, dans l’entreprise décrite par ce jeu de données, les emplyés soient avant tout sensibles à l’intérêt de leur projet et à leur charge de travail, ce qui bien sûr conditionne aussi leur niveau de récompense et de bien-être.

B. Happiness at work

|  |
| --- |
| **But**   * Prédire la satisfaction d’un employé: tâche de régression en machine learning. * Monitoring : A partir du profil d’un employé, faire apparaître des alarmes si certains indicateurs sont dans le rouge. * Conseiller à responsable des ressources humaines des plans d’amélioration du bien-être de l’employé concerné. |

1. Travail préparatoire

Les étéapes d’exploration des données, cleaning, preprocessing et feauture engineering sont issues du travail fait dans la partie Turnover prediction. En effet, nous utilisons toujours le premier jeu de données : “human resources”. Le travail fait en amont est donc réutilisé dans cette partie. Nous n’avons pas créé de nouvelles features par rapport au cas où nous souhaitions prédire la colonne “left”.

1. Prédiction de la satisfaction

*Training*

Nous voulons prédire le vecteur “satisfaction” à partir de toutes les colonnes sauf “left”, en partant du principe que le responsable RH ne sait pas si l’employé souhaite quitter l’entreprise à court terme ou non.

Nous avons testé plusieurs algorithmes dont

La fonction coût que nous avons choisie est l’erreur des moindres carrés (*Root Mean Square Error* RMSE).

*Choix de l’algorithme*

C’est Random Forest qui a obtenu le meilleur score id est le RMSE le plus bas sur le training set.

|  |  |
| --- | --- |
| Mean Squared Error (MSE)Average of the squares of the errors | 0.032866 |
| Root Mean Squared Error (RMSE)Root of the above mesure | 0.18129 |

*Testing*

On obtient les performances suivantes pour Random Forest:

* mse=0,0291
* rmse=0,1707

L’importance relative des variables dans la régression est présentée par le graphe suivant. On remarque que la colonne “investment” obtenue en multipliant average\_monthly\_hours par number\_project. Encore une fois, c’est la charge de travail et l’intérêt de l‘employé pour ses projets qui détermine la satisfaction d’un collaborateur.

NB: statistiques obtenues avec Random Forest sur le training set.

3. Warnings

fonctions warnings\_input et warnings\_output

concept: à partir des infos sur un employé, dire s’il travaille trop, risque d’être insatisfait, risque de partir, risque de s’ennuyer...

input = on rentre les paramètres pour renseigner le profil employé

output= messages du genre green light / red light / petit message de warning

Paramétrable par l’entreprise en fonction des normes du secteur et des habitudes de travail du

Use case : les RH demandent aux employés de remplir un formulaire en ligne à titre indicatif pour faire des statistiques. Pas d’anonymisation à ce stade.

C. Recrutement

|  |
| --- |
| **But**   * Recruter les meilleurs profils en fonction des besoins de l’entreprise. * Mobilité interne au sein de l’entreprise |

1. Exploration des données

Nous utilisons le deuxième jeu de données: Google Jobs. Les colonnes sont les suivantes: Company: Google à chaque fois

Title: intitulé du poste

Category: département, type d’activité

Location: Ville et pays d'implantation

Responsibilities: responsabilités associées au poste

Minimum Qualifications: compétences et/ou expérience minimales exigées

Preferred Qualifications: compétences et/ou expériences aussi appréciées

Cette fois-ci nous avons affaire à des données purement sémantiques.

2. Cleaning / preprocessing

Pour la colonne « Location » nous avons séparé chaque élément en deux, pour donner de nouvelles colonnes : « Town » et « Country ».

Nous avons décidé d’appliquer un preprocess différent pour les colonnes contenant des données sous forme de phrases, comme pour les 3 dernières : « Responsibilities », « Minimum Qualifications », et « Preffered Qualifications ». Nous avons commencé par isolé chaque mot, pour mettre les phrases sous une forme plus simple à analyser par la suite.

3. Feature engineering

L’objectif est de pouvoir appliquer des algorithmes de Machine Learning sur les colonnes, donc définir des distances entre les descriptions des responsabilités relatives à un poste, par exemple.

Pour les 3 dernières colonnes, « Responsibilities », « Minimum Qualifications », et « Preffered Qualifications », après avoir isolé les mots, nous avons supprimé les mots de liaison, et gardé seulement les racines des mots. Nous avons donc isolé au maximum le sens, pour chaque groupe de phrases.

4. Training

L’objectif était de pouvoir faire le tri parmi tous les jobs offerts par Google pour les regrouper et savoir quelles compétences sont les métiers qui vont ensemble dans la logique des compétences attendues par Google, pour orienter le demandeur d’emploi en le conseillant mais aussi pour permettre aux recruteurs de savoir quelles compétences vont ensemble pour une meilleure précision au niveau du recrutement. Donc nous avons appliqué un algorithme de clustering, pour former des clusters sur la base des responsabilités du poste, des attendus minimum et préférés pour le recrutement. Pour cela nous avons dû créer une distance entre 2 listes de mots, et pour cela nous avons utilisé deux distances :

Une distance créée par nous-mêmes, égale à :

N1 = nombre total de mots de la liste 1 ; N2 = nombre total de mots de la liste 2 ; MC = mots en commun : la distance est 1 – (N1-MC)\*(N2-MC)/(N1\*N2)

Nous avons aussi utilisé une distance calculée grâce à la bibliothèque python Word2Vec, qui calcule la distance entre deux mots en fonction de l’apparition simultanée de ces mots dans des milliers de textes. Cela prend en compte le sens des mots, et les liens que les mots ont les uns avec les autres.

On peut utiliser une fonction transformant un mot en vecteur, après avoir créé un modèle en fonction d’un vocabulaire (l’ensemble des mots utilisés, ici l’ensemble des mots apparaissant dans la colonne « Responsibilities » par exemple). Cela permet ensuite d’utiliser les distances classiques entre vecteurs.

5. Validation / choix de l’algo

Nous avons utilisé un algorithme de clustering (KMeans). Nous avons obtenu 20 clusters, compromis entre le temps d’exécution et la meilleure minimisation de la distance intra-clusters et la maximisation de la distance inter-clusters. Le clustering s’est fait à partir de la colonne ‘Responsibilities’.

6. Testing, performances finales

D. Expériences apprenantes

|  |
| --- |
| **But**   * à compléter (voir modèle plus haut) |

1. Exploration des données:

Nous avons tout d’abord commencé par chercher les significations des 228 titres de colonnes, communément appelées headers, contenus dans les données issues du Kaggle Survey [3]. N’ayant pas accès à un dictionnaire des données, nous avons essayé de décrypter ces données en nous aidant des headers et du contenu des colonnes associées. Ensuite, nous avons fait des analyses pour voir quelles colonnes sont le moins remplies et donc inutilisables et lesquelles sont les plus porteuses de données.

Voici l’état initial des données avec 228 lignes non préparées :

1. Cleaning / preprocessing et Feature Engineering:

Une fois l’exploration des données réalisée, nous avons utilisés ce qu’il en est sorti pour sélectionner les colonnes ou ce que l’on appelle plus souvent features que nous avons conservés. Cette analyse a permis de réduire le nombre de features de 228 à 24, qui sont donc les colonnes les plus denses et les plus pertinentes pour notre application. Nous avons gardé dans un premier temps uniquement les colonnes qui avaient un taux de remplissage supérieur à 70%. Dans le cas contraire, les colonnes contenant beaucoup de vide auraient simplement biaisé notre analyse en ajoutant une dimension qui n’est pas présente pour le reste des données. Certaines des colonnes du dataset préprocessé sont des colonnes originelles du dataset (a)d’autres des colonnes que nous avons transformés en valeurs numériques en créant une échelle (b), d’autres sont des colonnes que nous avons ajoutés manuellement (c), et d’autres sont des colonnes que nous avons créées combinant plusieurs colonnes pour en extraire un scoring sur un aspect qui nous intéresse (d):

* **ID** (c): permet d'identifier permettant de différencier plusieurs personnes sans avoir accès à leurs informations personnelles.
* **Gender** (b) : Sexe de la personne, non-binaire ou abstention de répondre. (échelle 0-2)
* **Âge** (a) : l’âge de la personne au moment de répondre au questionnaire.
* **EmployementStatus** (b) : permet de savoir si la personne travaille à temps plein, à temps partiel ou ne travaille pas du tout. Aussi permet de savoir si la personnes est en recherche d’emploi ou non. (échelle 0-5)
* **CodeWriter** (b) : si la personne fait du code ou non. (0 ou 1)
* **FormalEducation** (b) : Quel niveau d’éducation (niveau de diplôme) ont ils reçus(échelle 0-5)
* **MajorSelect** (b) : Quelle est la matière dominante de leur Bachelor (1-10)
* **Tenure** (b) : depuis combien de temps ils font du data analysis (0-5)
* **FirstTrainingSelect** (b) : Utilité moyenne attribuée par tous les utilisateurs à la méthode qui à été la première utilisée par cet individu pour apprendre le data analysis
* **ParentsEducation** (b) : Niveau d’éducation le plus haut atteint par un des parents (échelle 0-7)
* **EmployerIndustry** (b) : Dans quelle type d’industrie ils travaillent (échelle de 0 à 11 en fonction de la distance conceptuelle avec l’informatique et la technologie)
* **SalaryChange** (b) : Evolution du salaire dans les 3 dernières années (échelle de 0 à 5)
* **JobSatisfaction** (a) : Score attribué à l’intérêt de leur travail sur une échelle de 0 à 10)
* **Learning Method** (d): Méthode utilisée pour apprendre la data science (échelle de 1 à 8) il y a plusieurs lignes par personne pour chaque méthode utilisée par celle-ci
* **Usefulness** (d) : pour chaque méthode d’apprentissage, le produit entre le pourcentage de leur apprentissage tiré de cette méthode et l’utilité qu’ils estiment en avoir tiré (échelle de 0 à 300)
* **ScorePays** (d) : Proximité du pays aux Etats-Unis culturellement (de 0 pour éloigné à 1 pour identique) à partir du module gensim d’analyse sémantique de python.
* **NumberOfOldJobs** (c): Nombre de postes occupés dans les 10 dernières années (échelle de 0 à 14)

Nous avons donc mis toutes les données sous format numérique pour être à même de faire des clusters grâce aux fonctions que nous connaissons. Nous avons fait ceci tout en ayant conscience du risque de biais que nous introduisons en créant ou en choisissant les méthodes de création des échelles numériques correspondant aux notions théoriques de base. Par exemple nous avons utilisé un outil de NLP (natural language processing) ayant été pré entraîné par google sur des textes qui ont été sélectionnés avec un biais. De plus, certains mots composés peuvent être écrits de façon différente dans le module (avec un tiret un espace ou non ou avec une majuscule à différents endroits) et donc ne pas avoir la même distance avec le mot de référence. Le choix du pays de référence (les Etats-Unis) est aussi un biais culturel de notre part.

Il est également délicat d’attribuer des scores numériques au niveau d’éducation de personnes. Toutefois, nous gardons un aspect personnel dans notre preprocessing en mettant au coeur de nos données des scoring créés non pas à partir de notre appréciation personnelle mais à partir des avis donnés par les personnes elle mêmes sur leur travail et le niveau d’utilité de leurs différentes formations. C’est l’intérêt de ce data set. Malgré le travail supplémentaire de cleaning induit par un grand nombre de colonnes, la granularité importante des types d’informations disponibles permet d’avoir l’avis de personnes elles mêmes et d’avoir des données se rapprochant d’un CV commenté. Ceci est un avantage par rapport aux données de Pôle Emploi qui, bien que très denses en informations, n’avaient pas cette appréciation individuelle.

3. Training et Validation / choix de l’algo

A partir de ces données préparées,  nous avons cherché à appliquer plusieurs modèles d'entraînement des données pour réussir à prédire l’utilité estimée des méthodes d’apprentissages et quelle méthode d’apprentissage pourrait être la plus utile à un profil donné. On peut par exemple imaginer que dépendant du domaine d’étude choisi à l’université, une personne puisse être plus ou moins réceptive à des cours en ligne ou à l’apprentissage sur le lieu de travail. De même en fonction de l’âge ou de l’industrie dans laquelle la personne à choisi de travailler ou de bien d’autres facteurs.

Les résultats n’ont pas eu la précision attendue même s’ils restent très intéressants. Cela peut s’expliquer par un certain nombre de facteurs. Tout d'abord, la proximité des profils étudiés. Les gens qui ont répondu au sondage en ligne utilisé pour recueillir ces données sont tous issus du monde du data analysis, ils utilisent des outils similaires et ont des parcours similaires. Par ailleurs, le sondage était destiné aux utilisateurs de la plateforme Kaggle qui ont encore plus de similarités entre eux. Il est donc difficile d’obtenir la précision nécessaire de données pour les différencier de façon très nette par rapport aux profils proposés dans les données de pôle emploi qui étaient beaucoup plus éloignés entre eux. Par ailleurs, le passage des données lexicales à des données numériques implique la construction d’une métrique non seulement subjective mais aussi discrète et non continue. De plus, le nombre de champs possibles est souvent d’une dizaine, transformés en nombre cela ne permet pas une quantité suffisante pour s’approcher d’un comportement de données continues. Ainsi, il est très difficile

E.  Facteurs de succès

|  |
| --- |
| **But**   * à compléter (voir modèle plus haut) |