Rapport SD-TSIA 210
Sujet T3

V. Nacher-Castellet, Z. Gao, E. Gomez, C.Rodriguez et A. Sciberras
(Groupe 12)

Avril 2019

Table des matières

1 Introduction

Nous souhaitons réaliser une prédiction sur la localisation des réponses en fonction du contenu des réponses du grand débat et des contributions libres (sujet T3).

Nous avons d'abord pensé que nous focaliser sur le thème de la transition écologique était peut-être l'un des meilleurs moyens de prédire la localisation de la réponse car c'est un sujet dans lequel le lieu de vie a une grande influence.

Par ailleurs, nous pensons trouver peu de variations dans les réponses qui viennent de grandes villes françaises. Nous pensons que les habitants de Lyon et de Toulouse sont globalement affectés de la même manière par la transition écologique et les mesures qui sont prises à ce sujet en France. L'utilisation de la voiture dans des agglomérations à même densité de population nous semble être similaire, comme le sont les problématiques de pollution de l'air. Cependant, nous pensons que le regard sur la transition écologique peut être très différent dans une zone peu peuplée (et donc moins sujette à la pollution de l'air et moins bien desservie en transports).

Nous pensons donc que prédire la densité de la population de la zone de la réponse en fonction du contenu texte de celle-ci peut être intéressant et permettrait de mesurer les contrastes qu'il existe entre les zones urbaines et rurales, sur le thème de la transition écologique en France. Enfin, nous souhaitons réaliser une prédiction en fonction du contenu texte (comme demandé dans l'intitulé du sujet T3), donc nous étudierons plutôt les contributions que les questionnaires rapides.

Nous chercherons non seulement à trouver une prédiction qui nous donne un résultat satisfaisant, mais en plus nous aimerions comprendre quels sont les idées transmises dans le texte qui ont permis de réaliser cette prédiction. Un compromis entre une bonne prédiction et une bonne compréhension de chaque feature calculée est donc à trouver. Il nous semble, de plus, très difficile, sinon dénué de sens, de prédire une densité de population exacte. c'est pourquoi nous séparons les données en deux clusters :

- Le cluster "dense" correpond à des villes dont la densité de population est supérieure à 10 000 habitants par km^2. L'étiquette correspondante est 1.
- Le cluster "peu dense" correpond à des villes dont la densité de population est inférieure à 10 000 habitants par km^2. L'étiquette correspondante est -1.

Projet : **Prédire la densité de population du lieu de rédaction de la réponse en fonction du contenu textuel de celle-ci.** *NB : le code se trouve dans le notebook nommé "load data"*

2 Pré-traitement

La première étape du prétraitement consiste à lire le fichier .csv et choisir les colonnes utiles (questions, code postal et titre de chaque entrée) (cf. *load raw data*). On en fait une matrice où les lignes correspondent aux réponses au questionnaire et les colonnes correspondent aux questions. Puis on traite

chaque colonne selon sa catégorie. En effet, les questions à réponses binaires (cf. *preprocessing yesno*) sont traitées différemment que celles à choix multiple (cf. *preprocessing categories*) et que celles à réponse libre (cf. *preprocessing text*). Ce dernier traitement comporte le filtrage et le compte des mots. Pour ceci, on a utilisé les fonctions fournies dans le notebook (french-nltk.ipynb) de Christopher M. Church (cf. *get nltk text, no accents, filter stopwords,sort dictionary*). Finalement, on regroupe toutes les colonnes dans une seule matrice (cf. *preprocessing raw data*)

Une fois que les questions sont traitées, on calcule la vérité terrain. On cherche à différencier les grandes villes des petites villes en fonction de leur densité et on leur assigne une étiquette (cf *find zip codes by town, city village classifier*). Dans tout le projet, les grandes villes recevront l'étiquette "1" et les petites recevront "-1".

Dans un deuxième temps, on compte le nombre total de mots par question pour toutes les entrées (cf. *word count by question*). En choisissant de s'intéresser aux n mots les plus utilisés dans la question k par exemple, on peut obtenir le dictionnaire des mots des n mots les plus utilisés en réponse à la question k et leur nombre d'occurrences respectifs dans la fonction *get most used words*.

3 Méthode 1 : Extraction des features par mot les plus utilisés.

3.1 Extraction des features : description de la méthode et de la réflexion.

Nous avons pensé à plusieurs types de features que nous pouvons extraire, notamment grâce à un décompte des mots les plus utilisés. Aux vues des différents types de questions, plusieurs types de features sont possibles :

- Pour les questions fermées :
  - Questions Oui/Non : pour chaque réponse, la feature donne 1 pour une réponse oui, 0 si pas de réponse et -1 pour une réponse non. On obtient une feature par question.
  - QCM :la feature donne 0 si pas de réponse et un entier supérieur à zéro sinon. S'il y a n réponses possibles, la feature sera 1 si la personne a choisit la première réponse, 2 si elle a choisit la deuxième, n si elle a choisit la $n^{ième}$ etc. On obtient donc, comme précédemment une feature par question.

    Dans le code les question QCM et Oui/non sont pré-rentrées dans la liste *yes no questions*.

- Pour le titre et les questions qui demandent une réponse avec du texte : La fonction *extract features* possède l'argument *nbre words*. Cet argument (un entier positif) correspond au nombre de mots que l'on souhaite étudier par question. Pour un nombre de mots choisis par question égal à n, et pour les features extraites à partir de la question k :

1. On sélectionne les n mots les plus utilisés dans les réponses à la question k dans tout le dataset.
2. On compte, pour chaque réponse à la question k, le nombre d'occurrences de chacun des n mots précédemment sélectionnés.
3. Chaque nombre d'occurrence ainsi calculé correspond à une feature.
4. On obtient donc n features par question.

Nous avons, par la suite, testé l'importance de ces features à l'aide de l'outil *SelectKBest* de sklearn. Les mots qui revenaient dans beaucoup de questions étaient par exemple le mot 'transport' ou bien 'pollut' (on retrouvait simplement la racine du mot pollution le reste ayant été éliminé par le pré-traitement). Cependant, l'importance des features ainsi calculées est très faible (inférieure à 0.01 pour les questions avec du texte, cf figure 1), ces features ne sont donc pas satisfaisantes pour ce qui est du titre et des questions avec du texte. Nous avons donc décidé d'en calculer d'autres, afin d'avoir une importance des features plus satisfaisante. Certains mots ressortent parmi les plus utilisés pour plusieurs questions différentes : il peut être intéressant de regarder quels sont les mots les plus utilisés dans toutes les réponses, en prenant en compte toutes les questions à la fois. De plus, il parait approprié de prendre en compte le fait que certaines questions se suivent entre elles.

— Création de features à partir des mots les plus utilisés dans tout le texte des réponses (toutes questions comprises) : sur le même principe que précédemment pour les features crées à partir des questions à réponse texte, nous allons compter le nombre d'occurrences des mots les plus utilisés dans toutes les réponses de la base de données dans chacune des réponses.

1. Pour $nbre\_words = n$ mots sélectionnés, on sélectionne n mots les plus utilisés dans toutes les réponses.
2. Pour chaque réponse, on compte le nombre d'occurrences de chacun des n mots sélectionnés.
3. Le nombre d'occurrences de chaque mot correspond à une feature
4. On ajoute donc $n = nbre\_words$ features au total.

— Certaines question (les questions 1 et 2 par exemple) se suivent. La première des deux est une question en Oui/non et la seconde une question avec du texte. Il peut être intéressant de lier les deux réponses. Pour les question k-1 (question fermée en oui-non) et k (question avec champ de réponse libre) et n mots sélectionnés on réalise les opérations suivantes :

1. On sélectionne les n mots les plus utilisés dans toutes les réponses de la base de données à la question k.
2. On compte, pour chaque réponse à la question k, le nombre d'occurrences de chacun des n mots.
3. On calcule la valeur correspondant à la réponse en Oui/non : -1 pour 'non', 0 si pas de réponse et 1 pour une réponse 'oui'.

4. pour chaque nombre d'occurrences calculé à l'étape 2, on multiplie par la valeur trouvée à l'étape 3). (par exemple si la personne avait répondu 'non' à la question k-1 puis avait utilisé 3 fois le mot 'transport' dans la question k(en supposant que le mot transport est le plus utilisé de la question k), la feature résultante serait -3).
5. La valeur calculée pour chaque mot sélectionné parmi les n mots à l'étape 4) correspond à une feature. On obtient donc n features de plus par couple de question $k-1; k$

On remarque que les features ainsi calculées (particulièrement celles correspondant au nombre d'occurrence des mots les plus utilisés lorsque l'on prend les réponses de la totalité des questions en compte), ont une importance à peu près équivalente aux précédentes (les mots « transport » et commun ») ressortent particulièrement par exemple. Cependant, cette importance est toujours assez faible (inférieure à 0.01 pour $n = 10$ mots, voir figure 1 au paragraphe suivant).

3.2 Test des features calculées.

Nous avons effectué plusieurs tests sur ces premières features. Pour cela, nous avons utilisé la fonction *extract features* avec un nombre de mots égal à 10.

3.2.1 Test 1 : Importance des features

On utilise l'outil SelectKBest de sklearn pour calculer l'importance des features calculées au paragraphe précédent. On voit, comme expliqué au sous-paragraphe précédent que les features on une importance assez faible. Cependant, peut être qu'en utilisant la méthode de la Principale Component Analysis (PCA), nous parviendrons à avoir des features suffisamment importantes pour les différentes méthodes de machine learning vues en cours.

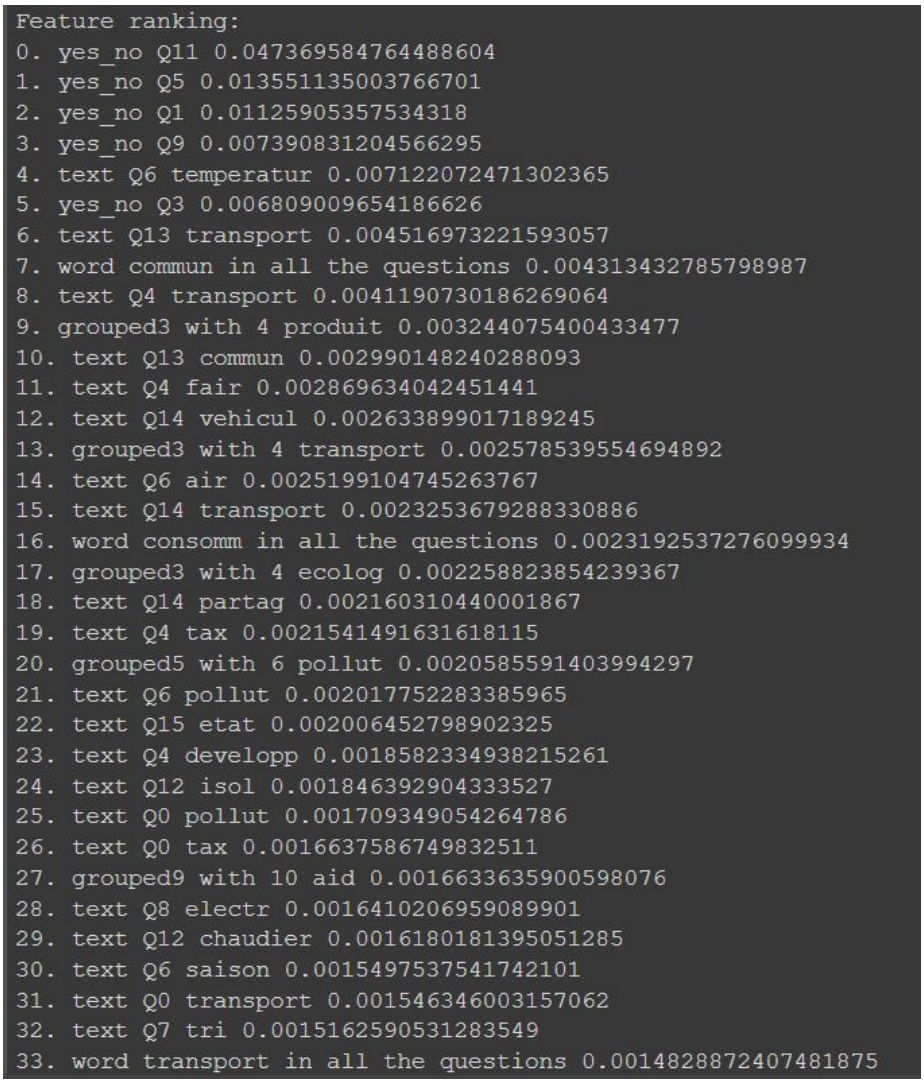

FIGURE 1 – Importance des features (la première réponse par texte est au rang 6)

L'importance des features calculées à partir des questions à réponse texte est faible : inférieure à 0.01 pour $n = 10$ mots.

3.2.2 Test 2 : Implémentation de méthodes de Machine Learning

On teste différentes méthodes sur les features calculées afin de tester leur efficacité : des méthodes d'ensembles (Arbres de décision, Forêts aléatoires, Bagging), ainsi que Support Vector Machines, Linear Discriminant analysis et Quadratic Discriminant Analysis. On utilise la validation croisée et la PCA pour avoir des résultats intéressants (Rappel : nous avons choisi les dix mots les plus fréquents pour l'extraction de features). Un total de 176 features ont été extraites.

Ensuite, devant le nombre de classifieurs qui attribuaient presque tous leurs labels dans la catégorie 'dense', on a choisit de ré-échantillonner les observations afin que le poids de la classe 'peu dense' soit à peu près le même que celui de la classe 'dense', avec plus ou moins de succès. Voici les résultats de ces classifieurs.

Decision Trees

— Done in 659.282s
— Average and std CV score : 0.8753308774969433 +- 0.001693514161190486,
— Score : 0.8779965047647311

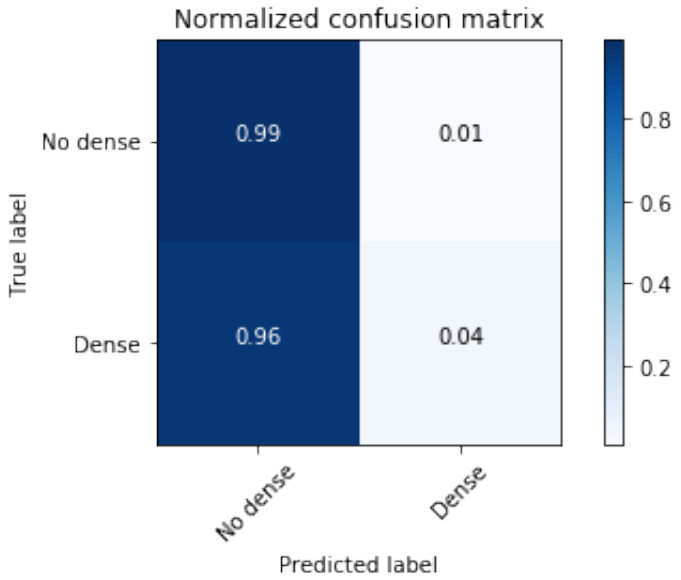

FIGURE 2 – Decision Trees matrice de confusion

Le score est plutôt bon mais l'on voit bien que le classifier a tenté de maximiser la pertinence des résultats en attribuant presque que des labels "peu dense", ce n'est pas satisfaisant.

Random Forest

— Done in 515.630s
— Score : 0.5213433918987095

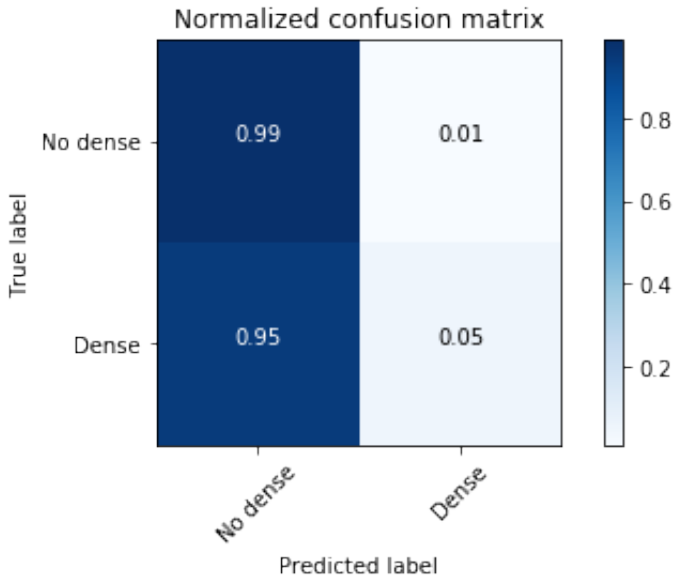

FIGURE 3 – Random Forest matrice de confusion

Le résultat est presque le même au niveau de la matrice de confusion que pour Decision Trees, mais avec un score moins élevé cette fois, le résultat n'est donc toujours pas convenable.

SVMs et Bagging

En raison de la grande quantité de données, le Bagging, le SVM linéaire et le SVM non linéaire ne peuvent pas converger à moins d'utiliser un nombre très important d'itérations. Nous continuons donc nos tests avec la Linear Discriminant Analysis.

LDA

— Done in 13.727s

— Average and std CV score : 0.876825103462964 +- 0.0014230985306608812

— Score : 0.8791835657994527

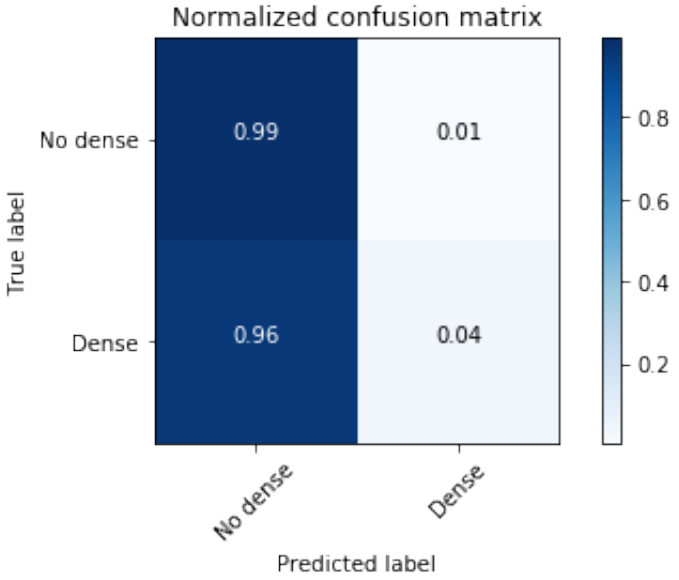

FIGURE 4 – LDA matrice de confusion

De même, bien que le score soit meilleur que pour les forêts aléatoires, on obtient une mauvaise matrice. Une explication possible est que LDA est un classifieur linéaire, il est difficile d'obtenir des résultats optimaux lorsque la corrélation de features est forte. Par conséquent, nous utiliserons PCA dans les paragraphes suivants avec LDA pour améliorer les résultats.

LDA avec PCA 20 components

— Done in 1.845s

— Average and std CV score : 0.5684184019086465 +- 0.0010100029795813673

— Score : 0.5956078741715303

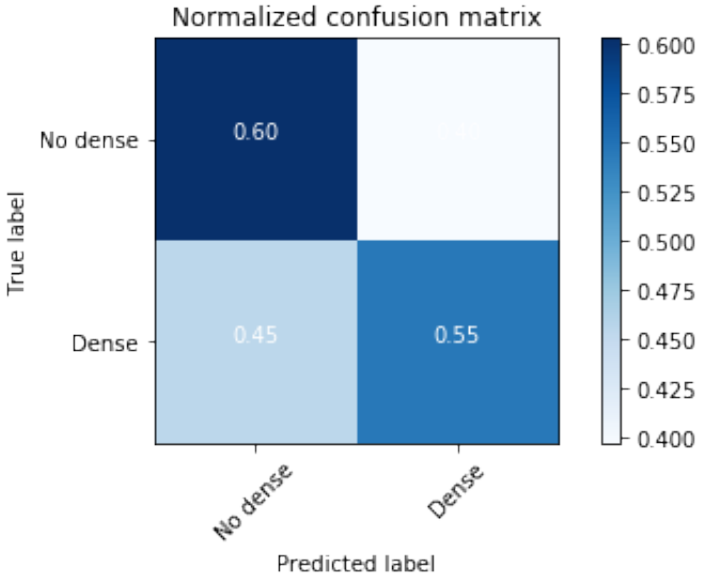

FIGURE 5 – LDA avec PCA 20 components

LDA avec PCA 30 components

— Done in 2.748s
— Average and std CV score : 0.5828139558661404 +- 0.002328466854591185
— Score : 0.599202031193326

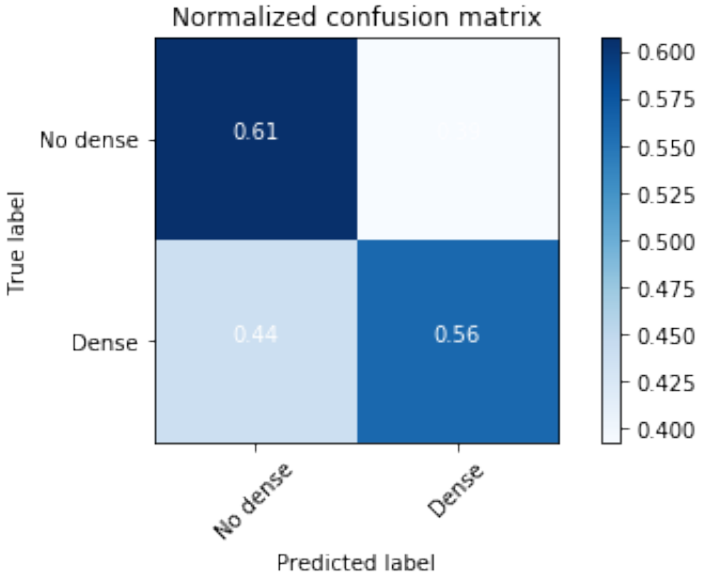

FIGURE 6 – LDA avec PCA 30 components

LDA avec PCA 40 components

— Done in 3.784s
— Average and std CV score : 0.5952587902513221 +- 0.0024317888991556205
— Score : 0.6176014772315098

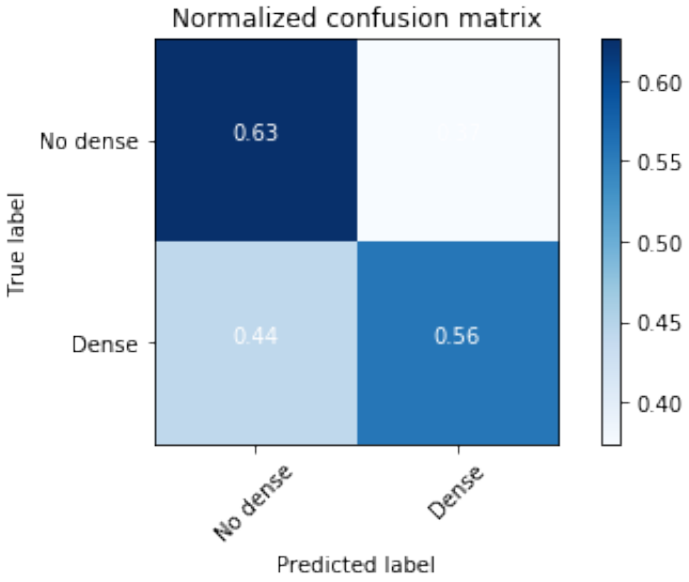

FIGURE 7 – LDA avec PCA 40 components

Le meilleur résultat est obtenu pour un nombre de composantes égal à 40, en effet le score est maximal et la matrice de confusion montre bien que les deux labels sont plutôt bien discriminés, malgré l'importance faible des features de texte démontrée au paragraphe précédent. De plus, en projetant sur 40 composantes, nous sommes sûrs que les features décrivant réponses aux questions ouvertes ont un rôle dans la prédiction. Bien qu'un score de 0.6 reste assez faible, cela semble acceptable compte tenu de l'importance des features. Comparons avec la QDA.

QDA

— Done in 8.681
— Average and std CV score : 0.3572308538704942 +- 0.07977955020280666
— Score : 0.35077653576021367

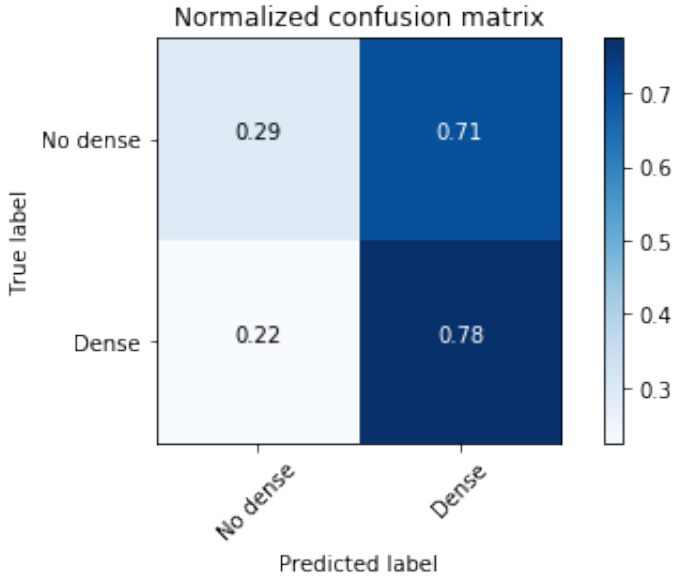

FIGURE 8 – QDA matrice de confusion

Cette matrice semble meilleure que les précédentes, avant de changer le nombre de composantes de la PCA, mais le score est très mauvais (inférieur à 0.5). On remarque cependant que la prédiction donne un bon taux de vrai positifs et de vrais négatifs par rapport aux faux positifs et aux faux négatifs. Nous allons tester cette méthode avec un nombre plus important de mots. Car contrairement à LDA, la QDA fonctionne mieux lorsque les données sont corrélées.

QDA avec 20 mots

— Done in 0.845s

— Average and std CV score : 0.6152025630048408 +- 0.22450615987090286

— Score : 0.467372308504

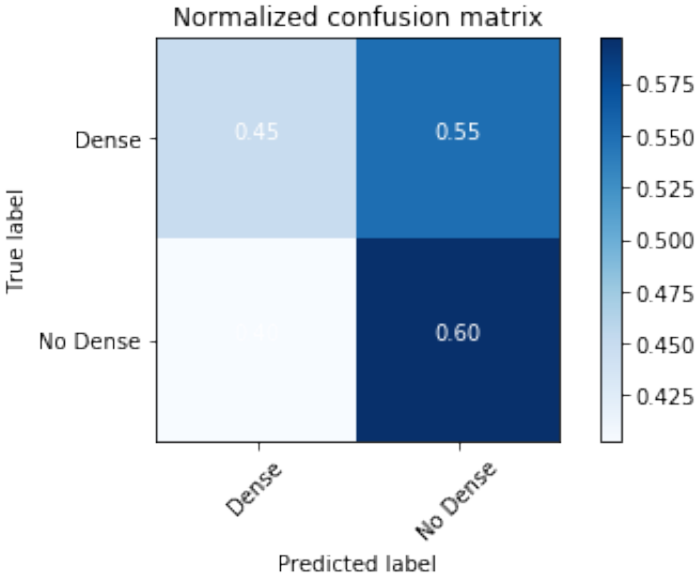

FIGURE 9 – QDA avec 20 mots

QDA avec 25 mots

— Done in 1.435s
— Average and std CV score : 0.3822311725473853 +- 0.24194137051579984
— Score : 0.388531671448

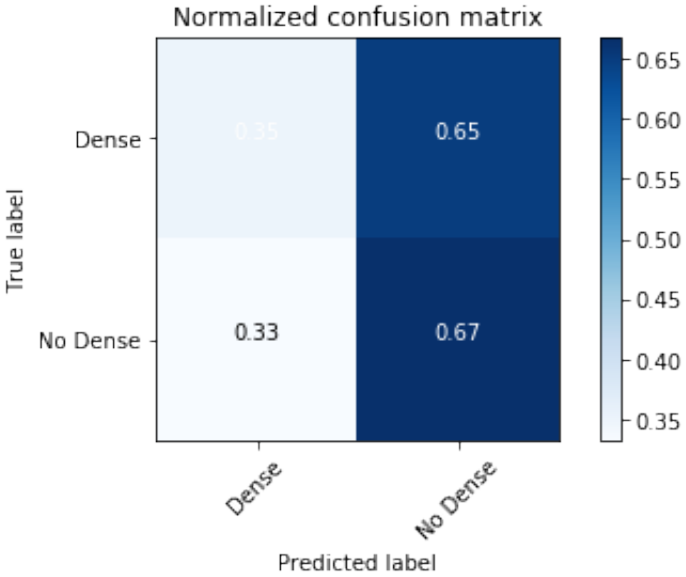

FIGURE 10 – QDA avec 25 mots

QDA avec 30 mots

— Done in 1.701s
— Average and std CV score : 0.3192386117827159 +- 0.2682827903789801
— Score : 0.347841857091

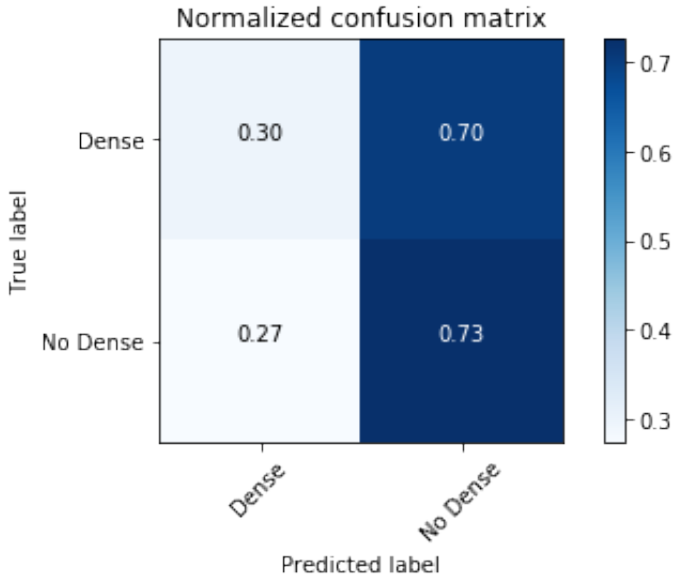

FIGURE 11 – QDA avec 30 mots

Le meilleur résultat pour la QDA est obtenu avec 20 mots à la place de 10 mais le score reste assez faible, et plus faible que la LDA, avec une matrice de confusion moins bonne.

La meilleure méthode semble donc être la Linear Discriminant Analysis avec un nombre de composantes égal à 40.

Conclusion de la méthode 1

La méthode 1 consiste simplement à regarder quels sont les mots les plus utilisés dans chacune des réponses en comptant le nombre d'occurrence pour chaque mot 'populaire' dans chacun des clusters. Elle donne des features de piètre importance pour les réponses aux questions ouvertes. Nous avons malgré tout essayé de prédire leur efficacité en testant plusieurs méthodes de Machine learning. Et, bien que les features soient peu importantes, elles permettent une assez bonne prédiction en utilisant une Analyse en Composantes Principales sur 40 composantes puis une Linear Discriminant Analysis. Cependant les features utilisés après projections perdent de leur sens, puisque qu'elles sont projetées dans la direction de plus grande variance (une feature est importante au sens de la PCA si elle maximise la variance). Ainsi, il nous est difficile de prédire quelle feature de texte est importante pour la prédiction, donc quel mot joue un rôle très important dans la prédiction de densité. On peut, néanmoins, obtenir les features les plus importantes au sens de PCA assez facilement. Compte

tenu du fait qu'un nombre important de composantes semble préférable, on en déduit qu'il y a beaucoup de features de petite importance qui, mises ensembles, permettent une bonne prédiction. La meilleure méthode pour avoir une idée des features les plus importantes pour la prédiction est celle des forêts aléatoires mais elle est d'une mauvaise efficacité. Devant un score d'*accuracy* qui plafonne à 0.6, nous avons testé d'autres méthodes d'extraction des features dans un paragraphe suivant.

Mais avant, nous avons choisit de donner une idée et une première analyse de cette extraction de features pour tenter de comprendre ce qui, dans les réponses et donc dans les différences de vies quotidiennes des personnes ayant répondu est différent sur ce thème lorsque la densité de population est très différente.

3.3 Premières visualisations de données grâce à la sélection des mots les plus utilisés

A partir de la fonction *features extraction*, on cherche simplement quels sont les mots les plus représentés dans toutes les réponses textuelles, dans chacun des deux clusters que nous avons définis. A partir des 100 mots les plus utilisés, on compare leur occurrence dans les deux clusters. Nous avons choisit, pour cette visualisation de représenter les mots qui ressortent le plus dans l'ensemble des réponses en texte de chacun des questionnaires. Nous ne conservons donc que les dernières features, celles qui comptent le nombre d'occurrences des mots les plus utilisés dans l'ensemble des questions.

— la classe -1 définit les communes de moins de 10000 habitants par km^2

— la classe 1 définit les communes de plus de 10000 habitants par km^2

Mots les plus utilisés pour le cluster 1 :

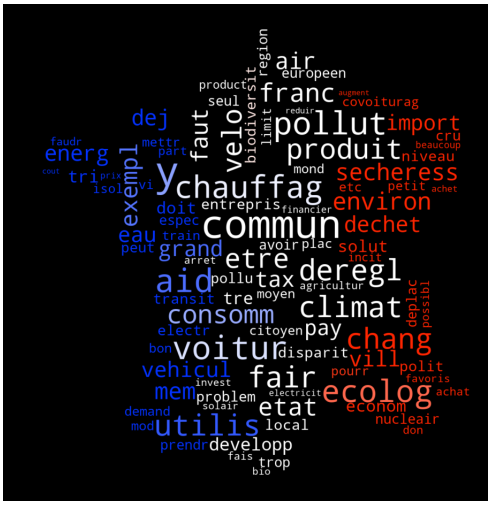

FIGURE 12 – Mots les plus utilisés pour une densité dans la classe 1 (densité haute)

Mots les plus utilisés pour le cluster -1 :

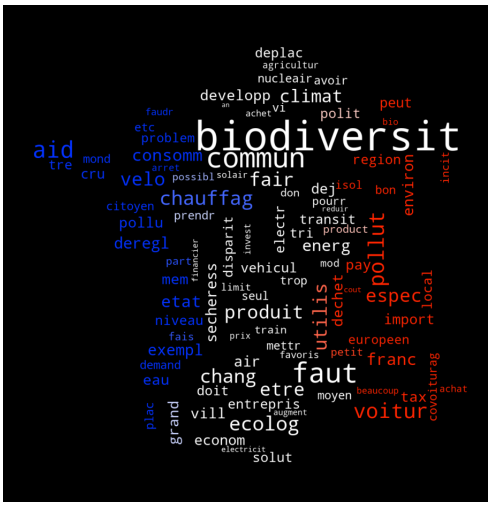

FIGURE 13 – Mots les plus utilisés pour une densité dans la classe -1

On peut tirer plusieurs déductions de l'analyse de ces deux schémas :

1. *Changements environnementaux*

   On remarque que les deux classes ne semblent pas touchées de la même manière par les changements environnementaux. Dans les zones plus denses, les mots "pollution" et "air", "voiture", "véhicules" ressortent bien plus que dans l'autre schéma, tandis que c'est le mot "biodiversité" qui bien plus important dans le schéma de la classe -1 que dans celui de la classe 1. Cela peut s'expliquer par le fait que les personnes vivant en ville ressentent plus la pollution de l'air, et l'impact de la voiture tandis que celles vivant à la campagne voient un changement dans la diversité des espèces qu'elles rencontrent dans leur vie quotidienne.

2. *Chauffage*

   L'utilisation du mot est plus importante dans les zones denses que dans les zones peu denses. Le chauffage a tendance à être centralisé et parfois difficilement changeable dans des immeubles, des appartements en location ou des copropriétés, ce qui peut être une explication. Il semble être une préoccupation moins importante dans les campagnes.

3. *État et échelons géographiques*

   Dans les zones denses, les mots "état" et "franc" (pour français, nous supposons) ressortent particulièrement, plus que dans les zones peu denses. Cependant le mot européen est présent dans la carte des zones peu denses tandis qu'il n'est que très petit dans celle des zones denses. Les deux cartes présentent le mot "monde" avec une taille relative qui est la même cependant. Peut-être est-ce parce que les zones denses voient l'importance de mesures au niveau national tandis que les zones rurales considèrent plus les mesures au niveau européen. Le mot "local" ressort également dans les deux clusters.

4. *taxes et achat*

On remarque beaucoup plus de mots mis en emphase sont liés à la consommation et aux achats dans le clusters à densité plus élevée, que dans celui à densité faible. Cependant les mots "tax" et "aides" ressortent beaucoup dans les deux images, de même que le mot "payer". Dans les deux cas, les mots en rapport avec l'argent et les finances semblent assez présent dans le débat sur la transition écologique.

5. *transports*
   Bien que le développement des transports varie beaucoup d'une zone à une autre, le thème semble être assez récurrent lorsque la transition écologique est évoquée (en témoigne l'utilisation du mot "commun" dans les schémas, qui vaut pour les mots "transports" et "transports en commun"), preuve qu'il peut être une alternative intéressante aux moyens de transports traditionnels, ou un sujet de tension pour les zones peu desservies.

Bien que simple, cette méthode de visualisation de données donne une bonne idée des sujets qui sont les plus importants dans les deux clusters étudiés et permet de noter les différences entre les zones denses et peu denses.

4 Méthode 2 : *tence2Vec*

4.1 Principe

Nous attribuons la performance limitée des méthodes présentées précédemment pour extraire les features des questions ouvertes au manque de contexte de l'utilisation du nombre d'occurrence des mots les plus utilisés. On perd une trop grande partie de l'information avec cette méthode : en effet, "A est plus grand que B" et "B est plus grand que A" ont les mêmes features. On doit donc utiliser une manière de représenter les données qui garde ce contexte, pour obtenir des features qui représentent mieux les réponses. Une méthode proposée pour le faire est *Sentence2Vec* qui est une extension de *Word2Vec* pour les phrases. L'objectif de *Word2Vec* est créer un vecteur de M dimensions qui code chaque mot dans un texte. Pour l'implémenter, on utilise un réseau de neurones comme Encoder. Cependant ce réseau calcule un seul mot et son contexte dans un vecteur, il nous est nécessaire d'étendre cela à des phrases de mots. Pour cela, on se contentera de prendre les vecteurs de chaque mot dans une phrase et de calculer la moyenne. On applique donc cette méthode à chaque réponse des questions de texte : ce que nous appelions phrase correspond ici à une réponse à une questions. Ainsi, on aura, pour chaque question et pour chaque réponse, un vecteur de M dimensions qui code le texte de la réponse. Finalement, on propose d'utiliser chaque dimension du vecteur comme une feature.

4.2 Test des features calculées

Aux vues du nombre de features générées par cette méthode, cette approche est très coûteuse. Il nous semble donc nécessaire de tester la méthode sur une petite portion des données : nous limitons à 1000 entrées la partie du dataset que nous utilisons pour les tests réalisés ci-dessous.

4.2.1 Test 1 : Importance des features

Comme réalisé précédemment pour déterminer l'importance des features calculées par la première méthode que nous avons présenté dans ce rapport, nous utilisons l'outil SelectKBest en classant les features par information mutuelle (cf figure 14). On constate que l'on obtient des features avec une information mutuelle plus grande que précédemment pour les features générées à partir du texte. Cette méthode semble donc meilleure pour trouver les features qui sont déterminée à partir des réponses constituées de texte.

```
Feature ranking:
1. feature 9 (0.108939)
2. feature 3 (0.042409)
3. feature 8 (0.033965)
4. feature 2 (0.011063)
5. feature 7 (0.001189)
6. feature 6 (0.000000)
7. feature 5 (0.000000)
8. feature 4 (0.000000)
9. feature 1 (0.000000)
10. feature 0 (0.000000)
```

FIGURE 14 – Ranking des features.

4.2.2 Test 2 : Test de la performance des features avec d'autres méthodes

Partant du principe que l'approche par Analyse Discriminante Quadratique fut la plus performante avec les features extraites avec la méthode 1 (au moment où nous avons réalisé ces tests), nous commençons par tester cette méthode sur le dataset de 1000 réponses. Le résultat est montré dans la figure 15.

— Done in 0.013s

— Average and std CV score : 0.4653 +- 0.0890

— Score : 0.54

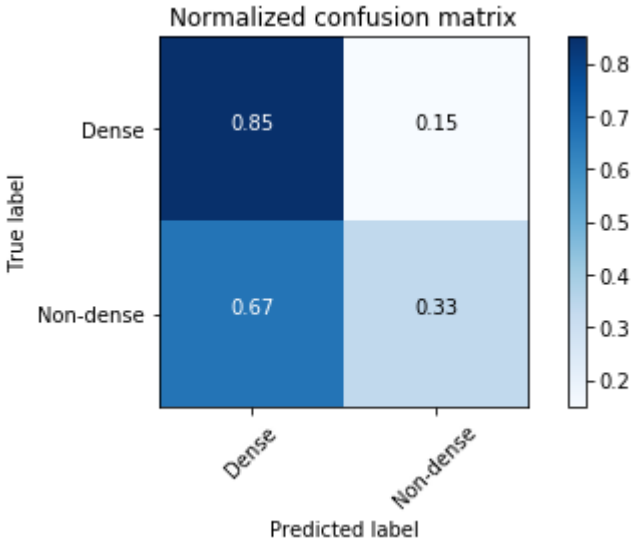

FIGURE 15 – Résultat pour la proposition de *ence2Vec*.

On constate une nette amélioration par rapport aux résultats précédents, avec un taux de faux positifs et de faux négatifs réduit (QDA sans PCA).

4.3 Conclusion de la méthode 2

Cette méthode d'extraction de features semble bien meilleure pour la prédiction puisque l'importance des features de texte et la QDA fonctionne bien mieux que lors de l'extraction des features avec la méthode 1. Cependant cette méthode, qui utilise des réseaux de neurones, ne permet pas de comprendre quelles sont les idées qui reviennent le plus dans les données de texte. En effet une feature correspond simplement à la ligne numéro i de la question donnée. Elles permettent simplement grâce à l'importance de voir quelles sont les questions qui permettent le mieux de différencier les deux clusters.

5 Conclusion générale

Nous souhaitions prédire la densité de population des communes des réponses des questionnaires sur la transition écologique du grand débat. Pour cela nous avons séparé les données en deux clusters (les villes de plus de 10 000 habitants au $km^(2)$ et celles de moins de 10 000 habitants au $km^(2)$).

Le but de cette prédiction est de comprendre ce qui, dans les réponses, est le plus déterminant pour prédire la densité de population de la commune depuis laquelle le questionnaire a été rédigé, et ainsi de comprendre les enjeux de la transition écologiques à la lumière des différences démographiques françaises.

Pour cela, nous avons extrait les features de deux manières différentes :

— Premièrement, nous nous sommes intéressés aux mots les plus utilisés dans la base de données et avons compté les différences d'occurrence de ces mots dans les deux clusters. Nous avons obtenu des features qui étaient facile-

ment compréhensibles (c'est à dire qu'elles permettaient de comprendre les différences entre les deux clusters), mais la perte d'information était trop grande pour obtenir une importance et une pertinence satisfaisantes. Cependant,une visualisation de données à partir de ces features nous a donné des pistes intéressantes concernant les clivages liés à la densité de population qu'il existe sur ce thème en France.

— En améliorant la prédiction grâce à une PCA, nous pensons qu'il est possible de garder une certaine part de compréhension des idées les plus clivantes, tout en améliorant la prédiction.

— Nous avons essayé une autre méthode d'extraction des features, qui converti chaque réponse en un vecteur en utilisant un réseau de neurone. Cette méthode d'extraction de features ne permet pas de comprendre, lorsque l'on voit l'importance des features, quelles sont les idées qui prédominent dans chacun des clusters et quelles sont celles qui les divisent. On peut seulement savoir quelles questions ont permis de différencier les deux clusters. La compréhension des enjeux est donc assez limitée avec cette méthode même si nous pensons que la prédiction est bonne.

On observe donc, ici, l'un des enjeux du Machine Learning et du Deep Learning. Tandis que l'un fait de moins bonnes prédictions, le second ne permet pas de comprendre facilement comment les prédictions sont réalisées. Une perspective intéressante ,qui permettrait peut être d'avoir une bonne prédiction tout en améliorant la compréhension, serait de générer des features avec la méthode *words2vect*, mais cette fois-ci en utilisant les mots les plus utilisés dans le dataset.

Références

[1] D. Adams, *The Hitchhiker's Guide to the Galaxy.* San Val, 1995. [Online]. Available : http ://books.google.com/books ?id=W-xMPgAACAAJ

[2] C. M. Church, "french-nltk.ipynb," u.d.

[3] U. Y. Nahm and R. J. Mooney, "Text mining with information extraction," in *Proceedings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases.* Stanford CA, 2002, pp. 60–67.

[4] C. Clavel, "Word embeddings. notes de cours. sd-tsia 214," 2019.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv :1301.3781*, 2013.

[6] R. Feldman and J. Sanger, *The text mining handbook : advanced approaches in analyzing unstructured data.* Cambridge university press, 2007.