Final report for SD210-Challenge Group 11

Authors: FANG Guyu, WANG Liang, WU Chun, Liang Guowei

I. Task definition (T3: Supervised learning for location, opinion, or any available specificity prediction) from the text content

By looking at the files about the four themes of the “Grand débat national”, we find that the available labels for supervised learning are the opinions obtained from yes/no questions and the zip code of authors. So, we divide our project to two main parts:

1) Predicting the opinion (answer to the close questions) using the propositions detailed in the open questions. This includes also examining which combination of answers yields the best results and which answer helps the most. This is a binary classification problem.
2) Predicting the location using the propositions of all questions. This is a multiclass classification problem.

Evaluation metric

For opinion prediction,

1) Accuracy. We compute the accuracy on the test set, but since the dataset might not be balanced. This may not be a good metric.
2) Precision, recall and F1-score.

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F_1 = \frac{2PR}{P + R}$$

For location prediction,

1) Accuracy
2) Confusion matrix.

II. Dataset description

For opinion prediction, we use “LA_TRANSITION_ECOLOGIQUE.csv” as dataset. It contains 99721 questionnaires. Each belongs to an independent author. The input is

the textual answers to certain open questions from a list predefined. The output is the answer (yes/no) to a specific question for opinion, for example, “Diriez-vous que votre vie quotidienne est aujourd'hui touchée par le changement climatique ?”. The potential bias could be that certain questions could be much more related to the target question than other questions are. Another potential bias could be that in some cases, there are much more people who have answered “yes” than people who have said “no”.

For location prediction, we have chosen “LA_FISCALITE_ET_LES_DEPENSES _PUBLIQUES.csv” as our dataset, because we think since people living in different regions of France have different income and different opinions about tax and social insurance, this theme is strongly related to the location. This file contains 126799 questionnaires. For input, we use all answers. For output, to simplify the problem, we aim to predict the region of the author instead of department. So, we first transform the zip code given by the dataset to the region code using the correspondence given by INSEE. Then, to further simplify the problem, we choose the five regions which have the most contributors. So, the output is one of the five regions’ code. The potential bias could be that different region has different population, so the dataset is unbalanced.

Data cleansing

Since not everyone has answered all the questions in the questionnaire, we have to eliminate the samples with unknown fields. The final number of available samples after data cleansing depends on the questions whose answers are considered to be input. Intuitively, the more answers we need, the less complete samples we have. Considering that our dataset is unbalanced in most cases and we have enough samples, we choose to under-sample the classes with more samples to balance the dataset.

Preprocessing

Punctuations removal and tokenization

For our task which is predict opinion and location, punctuations like “. , ' : ()” don’t bring much meanings. So, in order to reduce the dimension, we remove these punctuations. Then we replace all capital letters by lower case. We divide the texts to tokens which are the elementary units of text.

Stop words removal

Stop words have a grammatical function but do not carry much semantic information. As our task is a classification task, not a generation task, we remove the stop words to reduce dimension. Here, we use the stop word list defined by Veronis.

Stemming and lemmatization

In order to reduce dimension, we need stemming and lemmatization to Gather inflectional forms and derivationally related forms.

Stemming algorithms work by cutting off the end or the beginning of the word, taking into account a list of common prefixes and suffixes that can be found in an inflected word. In our project, we find that for French. It may not be very efficient in reducing dimension. For example, the word “avez”, after stemming, it’s not changed. For the conjugation in present tense of a verb, there could be six forms.

So, we try to use lemmatization which takes into consideration the morphological analysis of the words. For lemmatization, the problem is, to lemmatize a word properly, we need the annotation of part of speech. For example, when we lemmatize “avions”, if it’s a noun, then it becomes “avion”, if it’s a verb, it becomes “avoir”. Since we don’t have these annotations for now, we use the default lemmatization. Because there is no standard tool for French lemmatization for now, we use the code from others (https://github.com/ClaudeCoulombe/FrenchLefffLemmatizer).

In a nutshell, we do lemmatization first, and then stemming.

III. Data representation

Bag of words representation

This consists in representing each token by a one hot vector. We combine the textual answers of selected questions as a text. Each text is represented by the sum of the one hot vectors. Intuitively, each text is represented by the frequency of word. The feature dimension equals to the size of vocabulary. To reduce dimension, we set a minimum frequency for a word to be a feature.

Tf-idf representation

Words which appear in many texts usually carry less semantic information than those which appear in a few texts. Tf-idf representation takes this into consideration. It allows to reduce the weight of frequent tokens. But, still, these two representations are sparse and lose the information of positions of words.

$$TFIDF(w,d) = TF_{w,d}.IDF_{w,d}$$
$$= TF_{w,d}.\left(\left(\log_2 \frac{M}{DF_w}\right)\right)$$

IV. Further dimensionality reduction

Even after all the tricks we use to reduce the dimension (stemming, lemmatization, minimum frequency, etc.), we still get a feature vector of about 8000 dimensions (accurate value depends on the specific case). Due to curse of dimensionality, this could lead to bad results of classifiers. So, we try to further reduce the dimensionality.

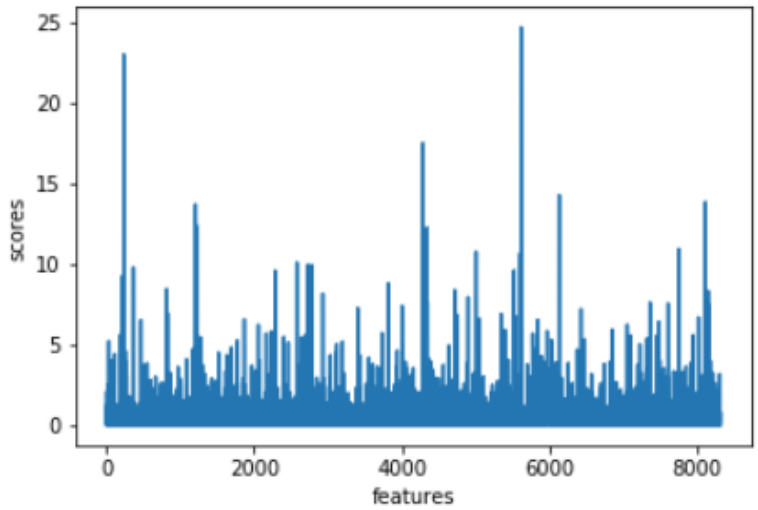
Principal component analysis

We first try to use PCA. After PCA, with 90% of variance retained, the dimension is reduced to about 2000.

Forward feature selection

Then we try to use the forward feature selection based on the Chi-squared stats of non-negative features for classification tasks. Here, empirically, we choose 2000 features. We will test the influence of the number of features chosen. By looking at the 10 features with highest scores, we verify that they are indeed related to the opinion (yes/no).

```
The 10 feautres with highest scores are: ['pollut' 'air' 'li' 'radical' 'économ' 'chang' 'chass' 'lobby' 'urgenc'
 'nucléair']
```



V. Opinion prediction

Naïve Bayes classifier

Naive Bayes classifiers are based on applying Bayes' theorem with strong independence

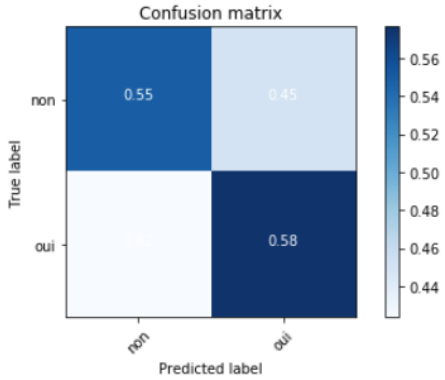
assumptions between the features. Here, we choose the Naïve Bayes classifier for multinomial models. It's suitable for classification with discrete features. But, in the library sklearn, it has been adapted to features like tf-idf. One characteristic of this kind of classifiers is that there isn't any hyperparameter to adjust. As we have balanced the dataset, the prior probability is a uniform prior. In practice, it's a very fast algorithm. The complexity in time doesn't increase a lot when the dimension of features increases.

We test this algorithm in three cases:

For output, we always aim to predict the answer to "Diriez-vous que votre vie quotidienne est aujourd'hui touchée par le changement climatique ? "

1) We take the answers to the first two questions "Quel est aujourd'hui pour vous le problème concret le plus important dans le domaine de l'environnement ?" and "Que faudrait-il faire selon vous pour apporter des réponses à ce problème ?" For us, human, these two questions are related semantically to the target. So, we think it may be easy for classifier.

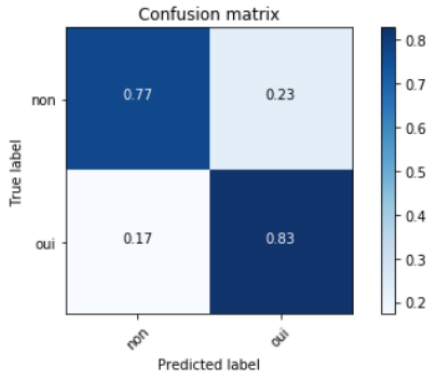
```
The accuracy is: 0.5629793510324483
The precision is: 0.5591320592039086
The recall is: 0.5765298562750037
The f1-score is: 0.5676976947767727
Normalized confusion matrix
[[0.54955219 0.45044781]
 [0.42347014 0.57652986]]
```



The results are not so good. They are just slightly better than random guess. We think this may be because that the features are not representative enough, or we need more textual data.

2) We use the answer to the question "Si oui, de quelle manière votre vie quotidienne est-elle touchée par le changement climatique ?" which is directly related to the target.

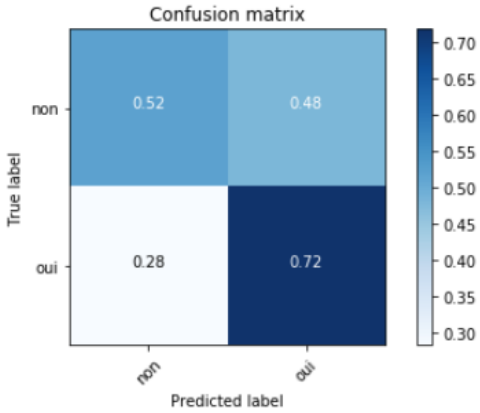
```
The accuracy is: 0.7978620019436345
The precision is: 0.768361581920904
The recall is: 0.8275862068965517
The f1-score is: 0.7968750000000001
Normalized confusion matrix
[[0.77052239 0.22947761]
 [0.17241379 0.82758621]]
```



The results are much better. This verifies that the answer to this question is closely related the target answer.

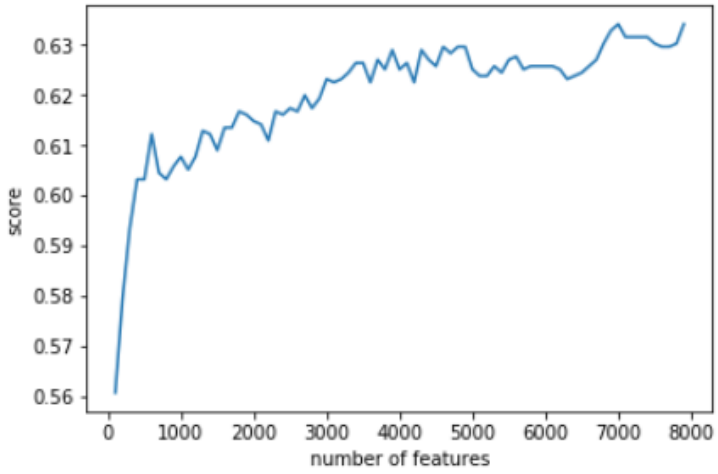
3) We use all the answers except the answers for the close question and the answer in the previous case to keep independence.

```
The accuracy is: 0.614790996784566
The precision is: 0.579004329004329
The recall is: 0.7181208053691275
The f1-score is: 0.6411024565608149
Normalized confusion matrix
[[0.51975309 0.48024691]
 [0.28187919 0.71812081]]
```



The result is better than in 1). The classifier is able to find most of the positive samples, but it is unable to classify negative samples as negative.

In this case, we also test the influence of the feature number. We can see that the score increases when the number of features increases. But it still makes sense to do forward feature selection to save time for some classifiers.



Random Forest

Model selection

Random forest combines the bagging and the random selection of features. Here, we tune two hyperparameters, number of estimators and the max depth using cross validation. We increase the number estimators to reduce variance. We limit the max depth to reduce overfitting.

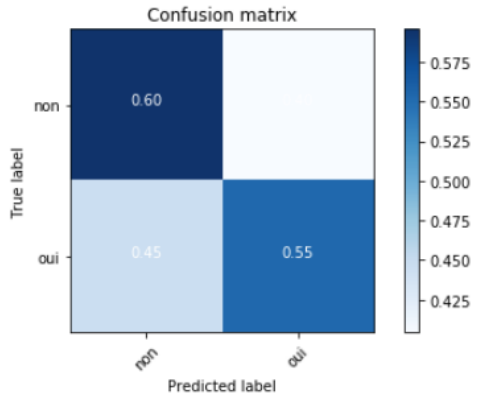
We find the best score and hyperparameters are:

```
Best Score: 0.6361736334405145
Best params: {'max_depth': 130, 'n_estimators': 130}
```

Test the previous three cases using the best hyperparameters:

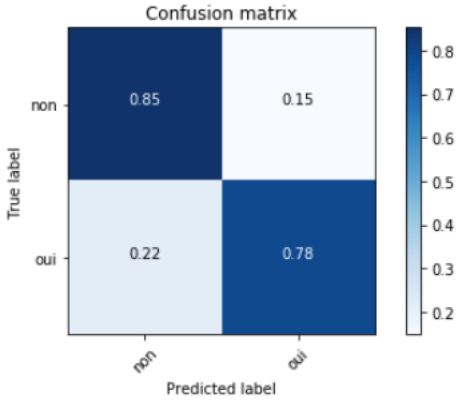
1)

```
The accuracy is: 0.5751474926253687
The precision is: 0.5760702186633816
The recall is: 0.5543043413839087
The f1-score is: 0.5649777240806464
Normalized confusion matrix
[[0.59580091 0.40419909]
 [0.44569566 0.55430434]]
```



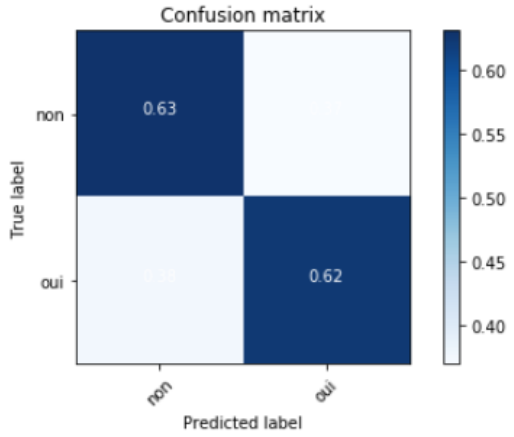
2)

```
The accuracy is: 0.8202137998056366
The precision is: 0.8304721030042919
The recall is: 0.7849898580121704
The f1-score is: 0.8070907194994786
Normalized confusion matrix
[[0.85261194 0.14738806]
 [0.21501014 0.78498986]]
```



3)

```
The accuracy is: 0.6276527331189711
The precision is: 0.6086387434554974
The recall is: 0.6241610738255033
The f1-score is: 0.6163021868787276
Normalized confusion matrix
[[0.6308642  0.3691358 ]
 [0.37583893 0.62416107]]
```



We can see that in all three cases, the results are slightly better than the results of naïve Bayes classifier. But the random forest takes more time to fit. And in the third class, the random forest has a higher precision and a smaller recall than the Naïve Bayes classifier.

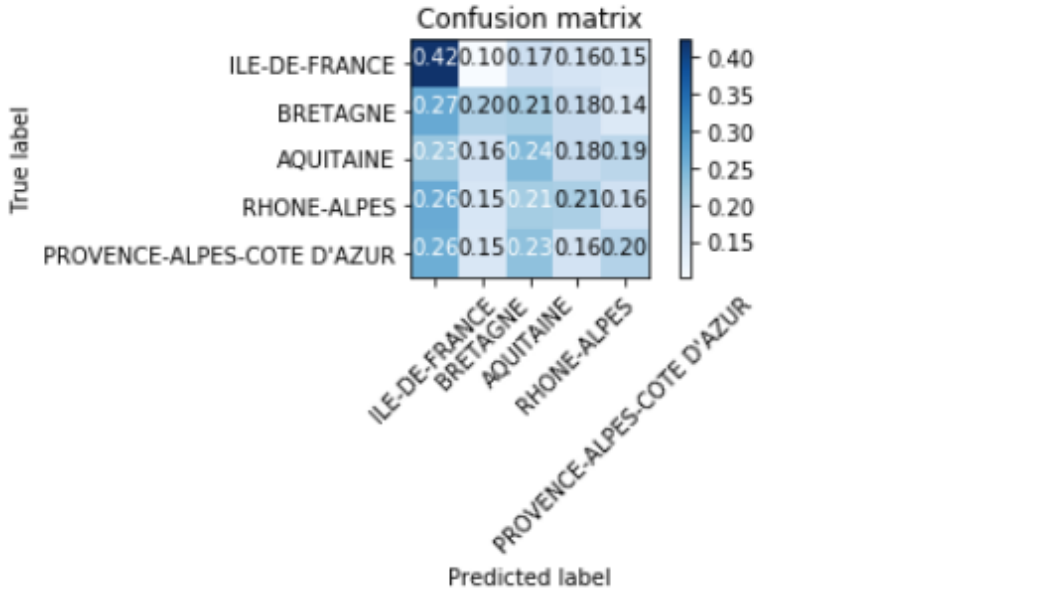
VI. Location prediction

Our goal is to predict the location in the following list using the whole questionnaire.

['ILE-DE-FRANCE', 'BRETAGNE', 'AQUITAINE', 'RHONE-ALPES', "PROVENCE-ALPES-COTE D'AZUR"]

Naïve Bayes classifier

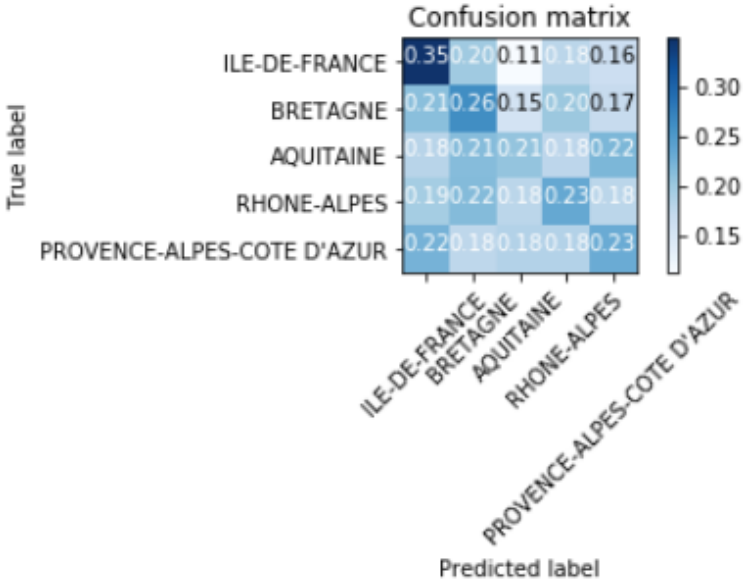
```
The accuracy is: 0.2546907574704656
Normalized confusion matrix
[[0.42287695 0.10225303 0.16984402 0.1559792  0.14904679]
 [0.26610169 0.20169492 0.21016949 0.1779661  0.1440678 ]
 [0.22535211 0.16021127 0.24295775 0.17957746 0.19190141]
 [0.26455026 0.15167549 0.21340388 0.20634921 0.16402116]
 [0.26041667 0.14756944 0.23263889 0.15972222 0.19965278]]
```



We can see that the average score is slightly better than random guess (0.2). The classifier is more able to classify “ILE-DE-FRANCE” than others.

Random forest

```
The accuracy is: 0.2567755385684503
Normalized confusion matrix
[[0.34835355 0.19757366 0.11265165 0.17850953 0.16291161]
 [0.21186441 0.26271186 0.15423729 0.20169492 0.16949153]
 [0.18133803 0.21478873 0.20598592 0.1778169  0.22007042]
 [0.19400353 0.21516755 0.17813051 0.2345679  0.17813051]
 [0.22395833 0.17708333 0.18402778 0.18402778 0.23090278]]
```



The results are similar to the results of Naïve Bayes classifier.

VII. Conclusion

By doing the preprocessing, tf-idf feature extraction and classification, we get some meaningful results. We can see that some textual answers are more related to the target than others.