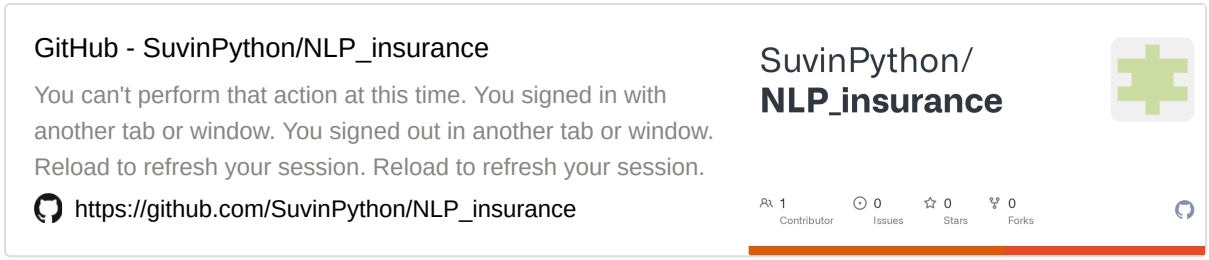
Projet 2 NLP - Insurance reviews

Team : Charlie Martin / Suvin Sasikumar (DIA 3)

Here is our link to our github, where you can find our project code



I. Exploratory Data Analysis

In this part we will present our study,visualizations and pre-processing about the dataset.

1. **Understand the dataset**

In this section, we will study some plots about the train dataset.

a) The train dataset look like this :

```
data_train.head(5)
```

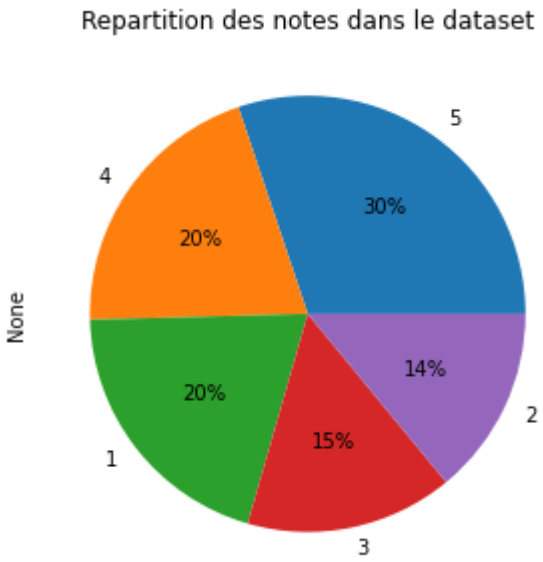
	date	note	auteur	avis	assureur	produit
0	06 septembre 2021 suite à une expérience en septembre 2021	5	brahim--k-131532	Meilleurs assurances, prix, solutions, écoute, rapidité, et je recommande cette compagnie pour vous \r\nDes prix attractif et services de qualité et rapidité	Direct Assurance	auto
1	03 mai 2021 suite à une expérience en mai 2021	4	bernard-g-112497	je suis globalement satisfait , sauf que vous avez un problème avec votre site internet ,impossible de déclarer un sinistre en ligne après plusieurs tentatives déclaration faite par téléphone ou tout c'est très bien passé , interlocutrice compétente et très agréable	Direct Assurance	auto
2	21 mars 2021 suite à une expérience en mars 2021	5	virginie-t-107352	Prix tres abordable plusieurs options s'offrent a nous comme le boitier connecter à la voiture, l'option tranquiliter et zero franchise ce qui est tout a fait plaisant	Direct Assurance	auto
3	10 juin 2021 suite à une expérience en juin 2021	4	boulain-f-116580	je satisfait du service, une réponse très rapide de votre service .je vous en remercie, vous êtes une assurance la moins cher sur le marche, Cordialement.	L'olivier Assurance	auto
4	29 janvier 2017 suite à une expérience en janvier 2017	1	ouaille31-51798	Client depuis plus de 25 ans, très déçu de cette "mutuelle" qui n'a plus rien d'une Mutuelle, la recherche du profit immédiat est devenu leur priorité. à l'agence on ne sait que essayer de vous fourguer des contrats inutiles, on vous fait payer une protection juridique sur chaque contrat mais vous ne serez défendu qu'une fois! dès qu'il y a le moindre sinistre il ne sont pas là pour vous défendre mais au contraire pour vous mettre d'office tous les tords pour faire descendre votre bonus qui est trop haut ! Bref ils n'ont plus qu'un seul but vous faire payer le plus possible. Hélas c'est maintenant le lot de tous les assureurs. Donc cherchez à payer le moins possible et surtout	Matmut	auto

We have 6 columns and 24105 rows.

Now, we are going to study some columns before doing any pre processing.

b) Column “note”

This column will probably become our target at the end of our project. To better understand the dataset, we studied the repartition of note in the dataset.



The 5-stars note are the most represent in the dataset (30% of the reviews) and the 2-stars note are the less present (14%). Globally, the dataset is equally distributed with all kind of reviews.

This is very usefull to train our futur model.

c) Column “assureur”

It is interesting to study if some insurance have more reviews than other.

assureur	
Direct Assurance	5896
L'olivier Assurance	4288
APRIL Moto	1023
GMF	998
Néoliane Santé	861

Top 5 of the most present insurance

assureur	
LCL	18
Mapa	10
Sma	6
MMA	4
Hiscox	1

Top 5 of the less present insurance

The repartition is not equal. We decided to keep insurance with more than 18 review, and drop the last ones.

d) Column “produit”

Here we will find a list of the insurance product, concerned about the review

produit	
auto	14077
sante	3525
moto	2105
habitation	1956
prevoyance	791
credit	653
vie	578
animaux	374
multirisque-professionnelle	20
garantie-decennale	12
assurances-professionnelles	8
responsabilite-civile-professionnelle	5
flotte-automobile	1

Repartition of the product in the dataset

With the same previous idea, we are going to keep review about product that are present more than 374 times;

2. **Pre processing**

After understanding our dataset, we applied some pre processing in order to have a cleaned dataset and to have column that will help us to build models.

a) we drop nan values

They are present only in the columns “avis’

```
for col in data_train.columns:
    zero = (data_train[col].isnull().sum() / len(data_train)) * 100
    zero = round(zero,4)
    print(f'    - {col} : {zero}%')
```

```
- date : 0.0%
- note : 0.0%
- auteur : 0.0041%
- avis : 0.0041%
- assureur : 0.0%
- produit : 0.0%
```

checking of nan values

b) Convert the column “date” format into “yyyy-mm-dd”

06 septembre 2021 suite à une expérience en septembre 2021

become ⇒ “2021-09-06”

c) Pre processing on the column “reviews”

We would like to do some analysis about the reviews. Before doing that, we need to clean this columns and keep only words. The idea is to have a list of words.

For example, we observe that a reviews looks like this :

avis
Meilleurs assurances, prix, solutions, écoute, rapidité, et je recommande cette compagnie pour vous \r\nDes prix attractif et services de qualité et rapidité

We created a list of only words. We dropped everything that are not words (like : number, special caracters)

We dropped words that are basic et non interstering for a reviews (like : et, je, ce, le, la…). We used the list of stop_words present in the ntlk.corpus libraries

After this pre processing, the previous reviews looks like this now :

avis
[meilleurs, assurances, prix, solutions, écoute, rapidité, recommande, cette, compagnie, prix, attractif, services, qualité, rapidité]

We can now analyse more deeply the reviews and create relation between insurance, product.

3. **Data Analysis**

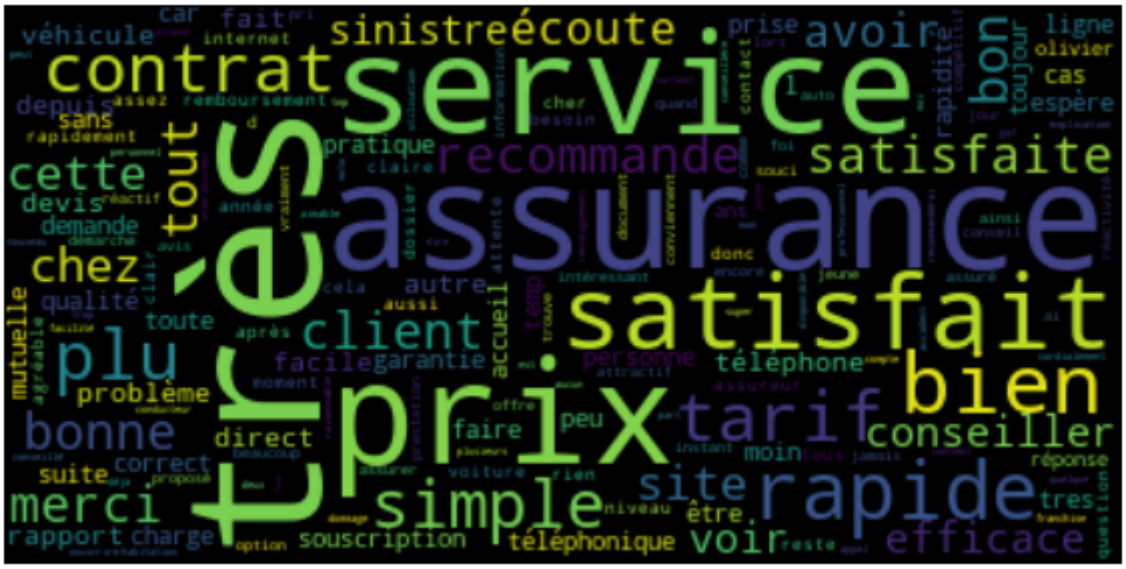
a) Wordcloud to visualize the frequency of the terms used in reviews

- Word cloud of the frequency of the word in review with rating 1

- Word cloud of the frequency of the word in review with rating 2

- Word cloud of the frequency of the word in review with rating 3

- Word cloud of the frequency of the word in review with rating 4

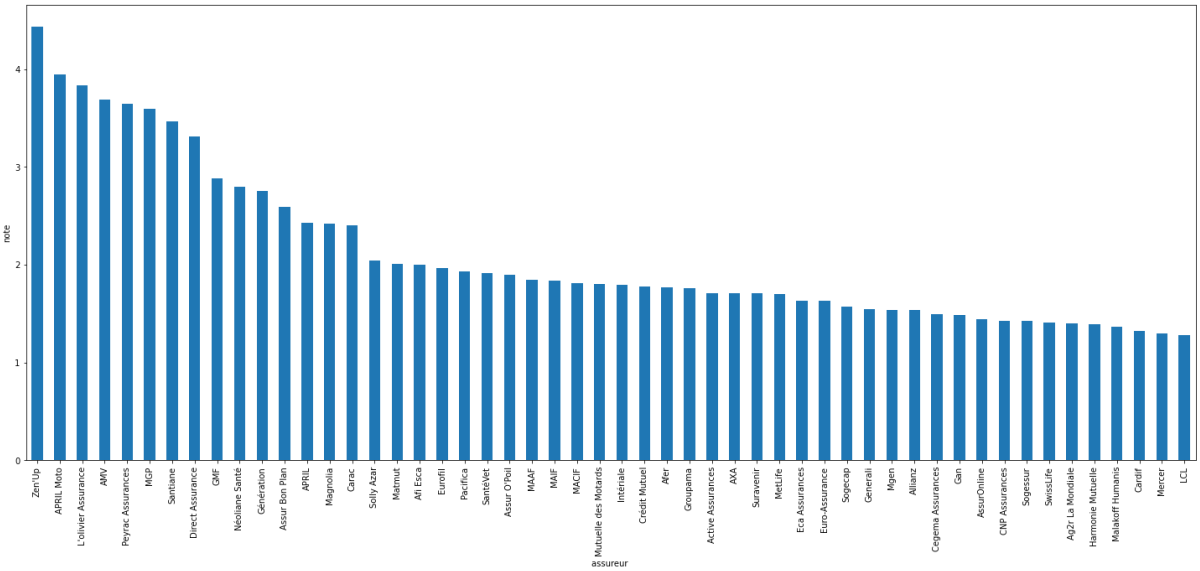


- Word cloud of the frequency of the word in review with rating 5

It is very intersting because the words characterize very well the note :

- note 5 = rapide, prix,merci, bien, recommande, satisfaire, ecoute, simple ⇒ only positive words
- note 1 = jamais, sinistre, aucun,.. we have no positive words. We have only neutral or some negative words

b) Average rating by insurance



Zen'up has the best average rating and LCL has the worst.

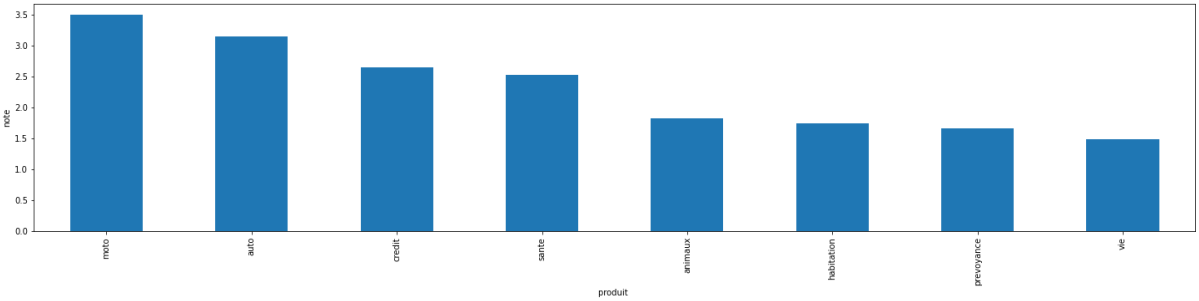
It can be intersting to visualize the world cloud of the frequency of terms used in the reviews for these two differents companies.

- Word cloud of the frequency of the word in reviews of Zen'up insurance

- Word cloud of the frequency of the word in reviews of LCL insurance

We have much more positive words in reviews for Zen'up (satisfait, service, efficace…), and we have some negative words for LCL (fuir, éviter, rien..)

c) Average notation by product



Moto and auto product are the one with the highest average rating .

Insurance "vie" are the one with the worst average rating .

We can study the wordcloud of the words for these two products.

- Word cloud of the frequency of word in the review for "auto" product

- Word cloud of the frequency of word in the review for “vie” insurance product

We have the same results as previsously with the insurance reviews. We have much more positive reviews in auto insurance than in life insurance

Conclusion on the exploratory data analysis:

We observed that the reviews explain very well the rating for a insurance. Insurance with good notation have much mure positive words in their reviews than insurance with bad rating .

But as we can see in the differents word cloud, some terms like “assurance”,”mois”,”contrat” are present in all rating. In this case, it could be interesting study the frequency of the same terms in every rating. We going to explain this in the next part and see the results.

Based on this obervation, we will need to build our model around the column “avis”, because the review will help us to answer our next two problematic : With the reviews, we can :

- **create unsupervised model to better understand the reviews and to create segmentations (clusters) of words**
- **create supervised model to predict the rating of a review**

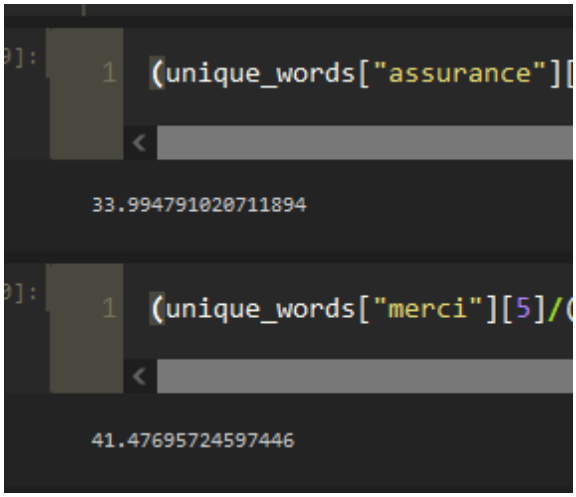
II. Unsupervised learning

The goal of this section is to create an unsupervised model to better understand the reviews. For that we will do a WordToVec model and use python librairies to add sentiments to our dataset.

How can we proceed ?

Initially, we wanted to remove words based on their frequency in the differents rating. If a word is well distributed among the all the 5 rating , then we remove it. This means to do a maximum frequency of presence in a note class of less than 35%.

Here is an example with two words: 'merci' and 'assurance'.



maximum frequency in one class for merci and assurance

```
['voir',
 'ete',
 'très',
 'trés',
 'professionnels',
 'journée',
 'prix',
 'véhicule',
 'fonction',
 'automobile',
 'regrette',
 'hauteur',
 'assistance',
 'aimable',
 'conseille',
 'assurer',
 'revanche',
 'professionnalisme',
 'ainsi',
 'trouve',
 'correspond',
 'c’est',
 'semble',
```

We can see that 40% of ‘merci’ are in the note 5. For ‘assurance’ the max frequency is for rating 1 and it is only 33%. Here you can see the list of our words ‘well distributed’. We remove these words in the new created column “avis2”.

After trying several models, we decided to improve this part in order to select more consistently and more precisely our words. That's why we used **TextBlob** here to get the sentimental value of the word, ranking from -1 for very negative to 1 for very positive (0 if neutral), including sentiments.

If the words have strong sentimental values (maximum value > 0.2), we keep them, otherwise they are removed in columns avis3 and avis4. Avis3 is the removal words from avis2 (already with a first manipulation on frequency rates) and avis4 is the removal from avis1 (without frequency rates).

	date	note	auteur	avis	assureur	produit	avis2	avis3	avis4
0	2021-08-06	5	brahim--k-131532	[rapidité, écoute, compagnie, prix, solutions,...	Direct Assurance	auto	[rapidité, écoute, compagnie, qualité, cette, ...	[qualité]	[qualité, meilleurs]
1	2021-05-03	4	bernard-g-112497	[ligne, internet, déclaration, bien, impossibl...	Direct Assurance	auto	[déclaration, impossible, tout, faite, télépho...	[impossible, tout, satisfait, sinistre, agréable]	[impossible, tout, satisfait, sinistre, agréable]
2	2021-03-21	5	virginie-t-107352	[tres, connecter, fait, prix, offrent, abordab...	Direct Assurance	auto	[tres, fait, abordable, franchise, options, vo...	[tout]	[plaisant, tout]
3	2021-06-10	4	boulain-f-116580	[moins, cordialement, remercie, assurance, trè...	L'olivier Assurance	auto	[moins, remercie, rapide, cher, réponse, satis...	[rapide, cher, satisfait]	[rapide, cher, satisfait]
4	2017-01-29	1	ouaille31-51798	[contrats, possible, défendu, besoin, client, ...	Matmut	auto	[contrats, possible, client, chaque, surtout, ...	[tous, trop, sinistre, déçu]	[tous, trop, inutiles, sinistre, immédiat, hél...

Here are the word vectors of the different manipulations. We can see that the best column is avis4: it only contains words with a sentimental value greater than that of avis3. This is what we will be able to see with the efficiency of the models. The best ones are also those generated from the words present in avis4. That is why we will present the results with the data from this manipulation.

WordToVec

Next, we are going to make a wordToVec model to create the unsepervised model. Like for the first project we created a WordToVec model based on the words :

```
model = Word2Vec(data_train['avis4'],vector_size=200, min_count=1,window=3)
```

```
w1 = "merci"
model.wv.most_similar(positive=w1)
```

```
[('facile', 0.9993181228637695),
 ('clair', 0.9990428686141968),
 ('agréable', 0.9989716410636902),
 ('claire', 0.99892258644104),
 ('amis', 0.9988728761672974),
 ('satisfaite', 0.9988531470298767),
 ('proches', 0.9988452196121216),
 ('sympathique', 0.9988067746162415),
 ('super', 0.998798668384552),
 ('correct', 0.9987568259239197)]
```

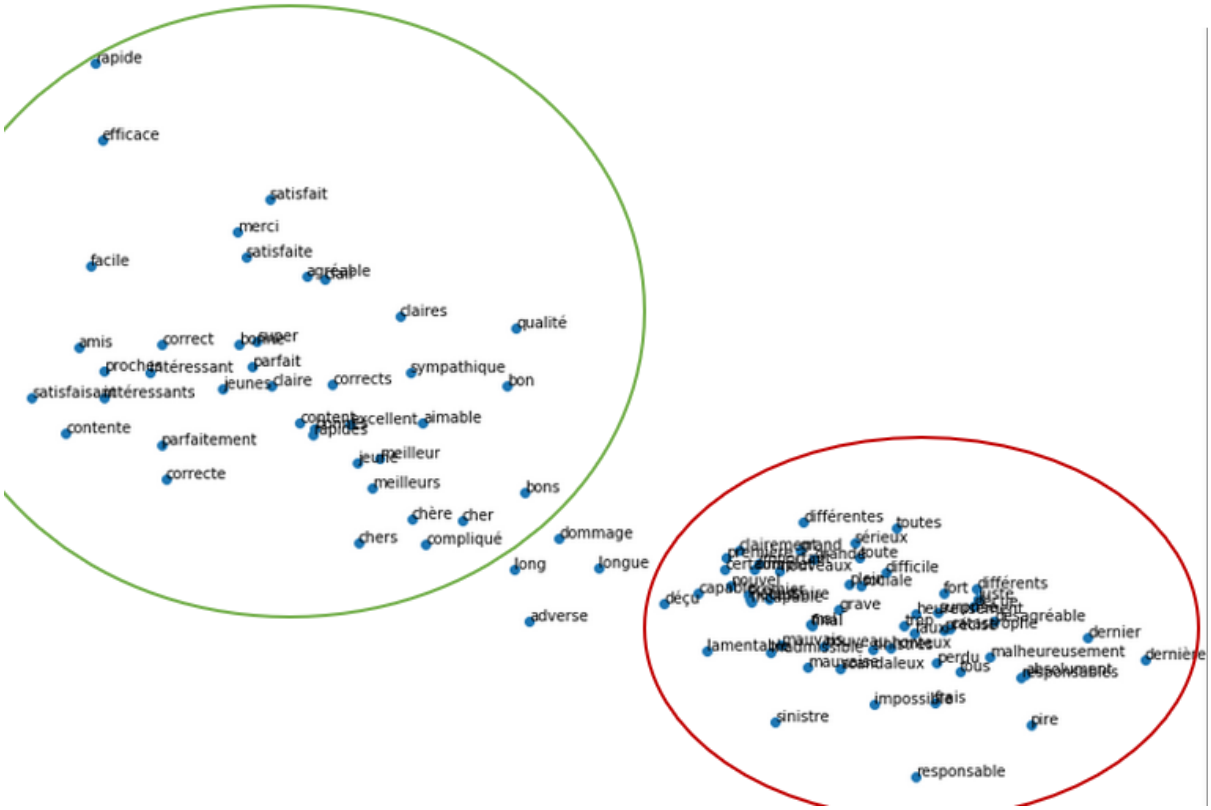
```
w2 = "mauvais"
model.wv.most_similar(positive=w2)
```

```
[('adverse', 0.9995973706245422),
 ('absolument', 0.9995914101600647),
 ('heureusement', 0.9995884299278259),
 ('dernier', 0.9995689392089844),
 ('honteux', 0.9995527267456055),
 ('propre', 0.9995526075363159),
 ('final', 0.9995512962341309),
 ('perdu', 0.9995465278625488),
 ('mauvaise', 0.9995401501655579),
 ('malheureusement', 0.9995397329330444)]
```

most similar words to merci

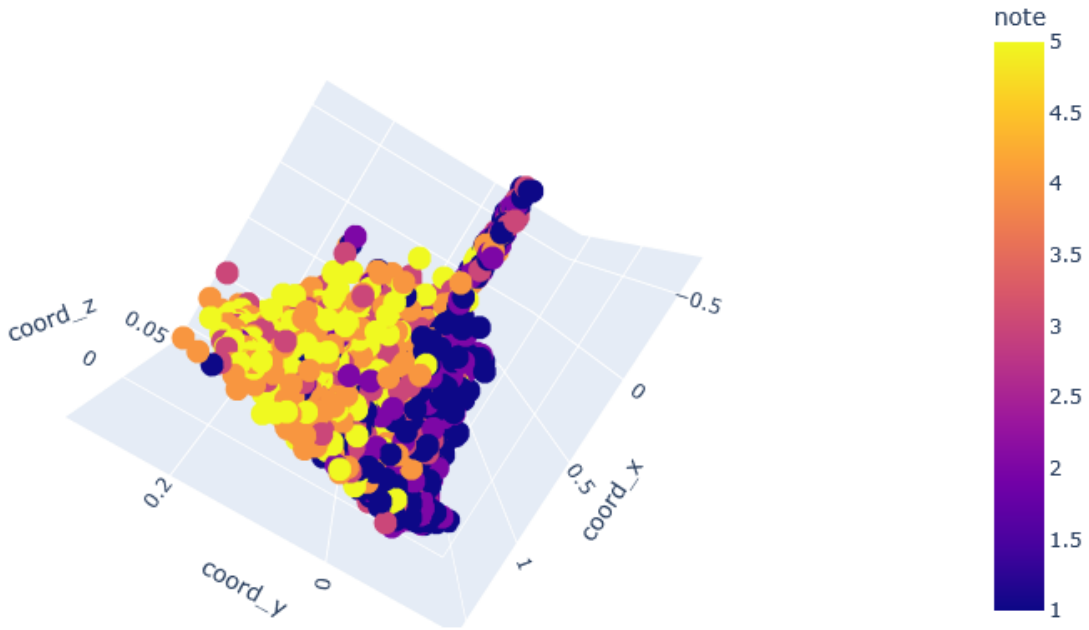
most similar words to mauvais

To see if the our WordToVec model is great we made some visualizations with a PCA and a plot :



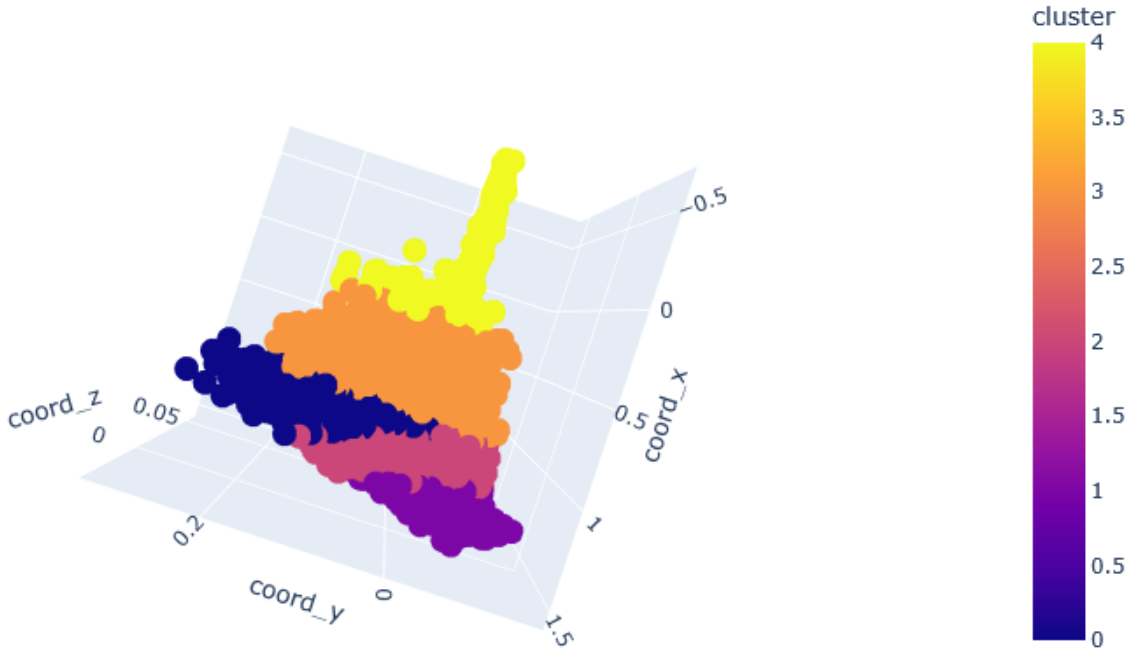
We can see that the positive words are in the green cluster and the negative ones in the red

The WordToVec model's sentiment is well separated between postive and negative words but here we can see that it is binary and for a 5 cluster models it will certainly not be enough. By the way we made another PCA to keep the 3 first axes (x,y and z) and have words coordinates.



scatter plot of the avis coordiantes with color representing the notes

We bring these coordinates into our dataset to a list of coordinates for each rows and we create a new column that is the mean coordinates for an ‘avis’. With this 3 coordinates mean we decided to create our first unsepervised model by making a kmeans with 5 clusters.

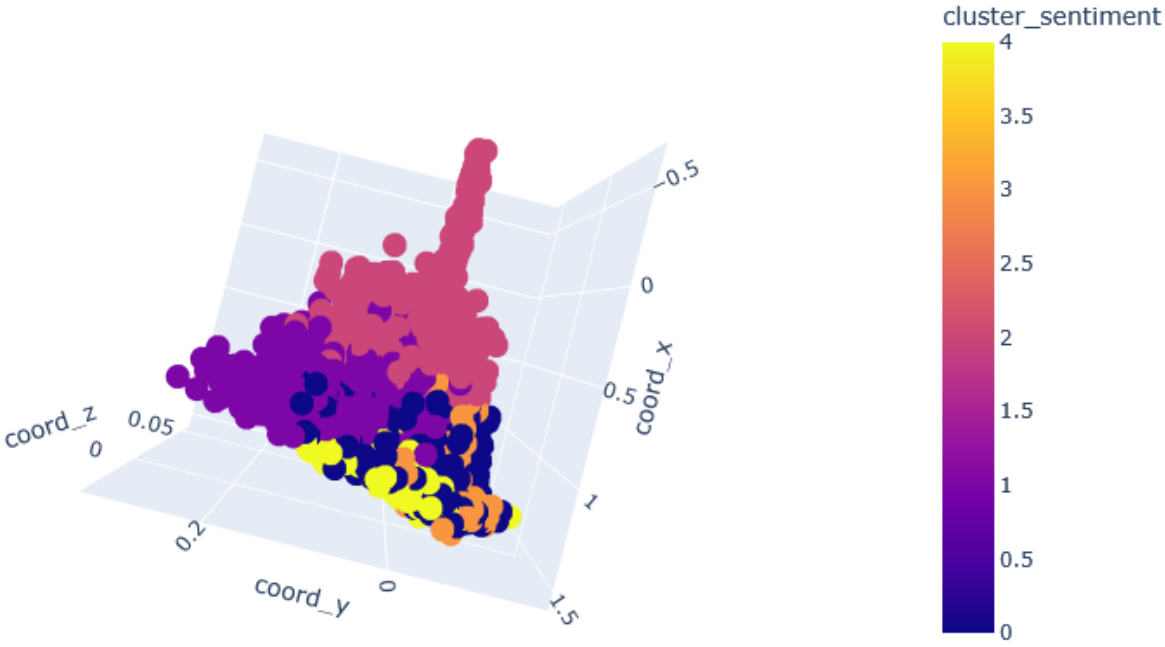


But this kmeans was not satisfaying and we decided to add a fourth value to our model : the sentimental value. Thanks to TextBlob library we calculated the mean sentimental value of the vector of words from ‘avis4’ column :

coord_x	coord_y	coord_z	sentiment
1.142804	0.087052	0.003838	0.465000
1.227246	0.025725	-0.000905	0.014000

our four label for unserpervised model

And we optain a new better kmeans :



It show us the importance of the global sentimental meaning of the sentence and the fact that in a great avis it can have some really bad words. Also, we can conclude with this unsepervised model that we have to create new variable for the supervised model because it will be not enough with just these ones.

III. Supervised

We are going to test different models to predict the rating (number of stars) of the review with the training dataset.

a) Pre processing

Before doing models, we have to do some pre processing on the train set.

We decided to create new features in order to have a better model to explain then predit the target (rating).

We created :

- number of words, based on the columns ‘avis’ because it is the initial column of the train_set, and we think that the number of words have a influence on the rating
- average word size, based on the column ‘avis’
- one hot encoding with the columns ‘assurance’ and ‘produit’, in order to have only numerical value
- We wanted to add a new column about the number of days since the problem happened and the day of reviews. As you can see below, we have a precise date for the reviews (’16 novembre 2021’) but for the day of the problem we only have the month (’novembre 2016’). So we didn’t keep this idea of feature.

```
16 novembre 2021 suite à une expérience en novembre 2021
16 novembre 2021 suite à une expérience en novembre 2021
 sinistres";L'olivier Assurance;auto
16 novembre 2021 suite à une expérience en novembre 2021
vier Assurance;auto
16 novembre 2021 suite à une expérience en novembre 2021
15 novembre 2021 suite à une expérience en novembre 2021
15 novembre 2021 suite à une expérience en novembre 2021
15 novembre 2021 suite à une expérience en novembre 2021

15 novembre 2021 suite à une expérience en novembre 2021
15 novembre 2021 suite à une expérience en novembre 2021
15 novembre 2021 suite à une expérience en novembre 2021
```

- We keep the the columns created in the unsupervised column :

	coord_x	coord_y	coord_z	sentiment
0	1.142804	0.087052	0.003838	0.465000
1	1.227246	0.025725	-0.000905	0.014000
2	0.511420	-0.019449	0.012000	0.410000
3	1.050377	0.184506	0.010466	0.146667
4	1.079747	-0.046210	-0.002646	-0.261429
...	...	...	...	...
'42	1.263016	-0.024110	0.005234	0.157500
'43	1.350888	-0.115577	-0.001333	-0.575000
'44	1.255296	-0.046054	0.002756	0.152000
'45	1.320290	-0.039519	0.019149	0.220000
'46	1.088309	0.049840	0.004839	0.583333

and after all the preprocessing, our train set looks like this now : we have a dataset of 21747 words and 66 columns

	coord_x	coord_y	coord_z	sentiment	longueur_moyenne	nombre_de_mots	AMV	APRIL	APRIL Moto	AXA	...	SwissLife	Zen'Up	animaux	auto	cred
0	1.142804	0.087052	0.003838	0.465000	0.204724	0.022177	0	0	0	0	...	0	0	0	1	
1	1.227246	0.025725	-0.000905	0.014000	0.190003	0.044355	0	0	0	0	...	0	0	0	1	
2	0.511420	-0.019449	0.012000	0.410000	0.154238	0.032258	0	0	0	0	...	0	0	0	1	
3	1.050377	0.184506	0.010466	0.146667	0.165354	0.020161	0	0	0	0	...	0	0	0	1	
4	1.079747	-0.046210	-0.002646	-0.261429	0.119798	0.110887	0	0	0	0	...	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21742	1.263016	-0.024110	0.005234	0.157500	0.157480	0.070565	0	0	0	0	...	0	0	0	0	
21743	1.350888	-0.115577	-0.001333	-0.575000	0.206693	0.030242	0	0	0	0	...	0	0	0	0	
21744	1.255296	-0.046054	0.002756	0.152000	0.158658	0.254032	0	0	0	0	...	0	0	0	0	
21745	1.320290	-0.039519	0.019149	0.220000	0.123604	0.084677	0	0	0	0	...	0	0	0	1	
21746	1.088309	0.049840	0.004839	0.583333	0.140343	0.066532	1	0	0	0	...	0	0	0	0	

21747 rows × 66 columns

Now we are able to construct our first models

We are using a test size of 33% of the train dataset, to train our models

```
from sklearn.model_selection import train_test_split
y = data_train["note"]
test_size = 0.33
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=test_size)
```

b) Supervised Model

In this section, we will analyse few models and their results. We decided to use the RMSE metrics and the matrice_confusion to study the results.

- **Linear Regression**

```
# Create a LinearRegression model
model = LinearRegression()

# Fit the model to the data
model.fit(X_train, Y_train)

# Use the model to make predictions
y_pred = model.predict(X_test)
```

and the results on the prediction :

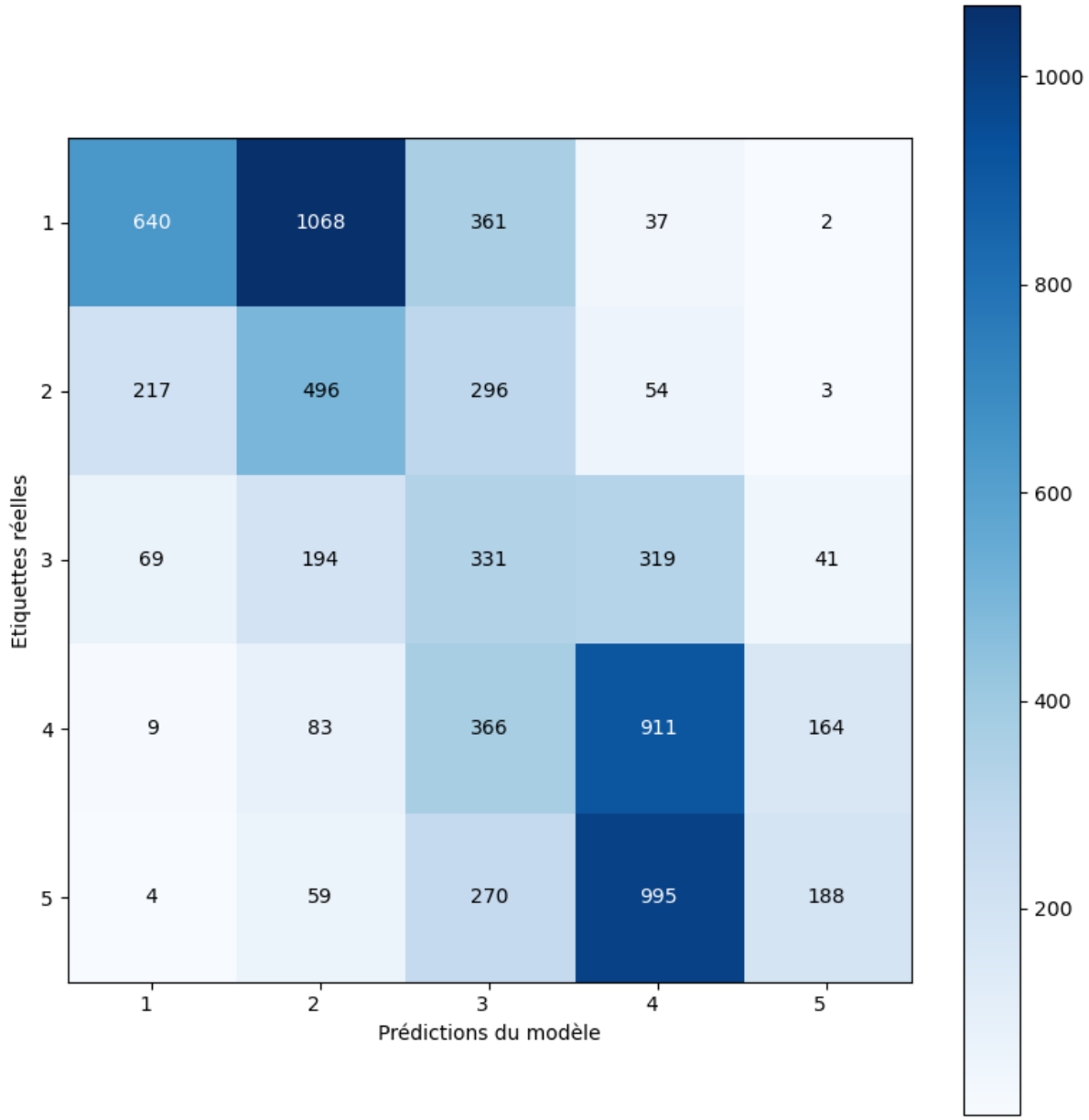
```
from sklearn.metrics import mean_absolute_error,mean_squared_error, r2_score
import math

MSE = mean_squared_error(Y_test, y_pred)
RMSE = math.sqrt(MSE)
mae = mean_absolute_error(Y_test, y_pred)
r2 = r2_score(Y_test, y_pred)
accuracy = model.score(X_test,Y_test)

print(f'Mean squared error: {MSE:.2f}')
print(f'Root Mean squared error: {RMSE:.2f}')
print(f'Mean absolute error: {mae:.2f}')
print(f'R2 score: {r2:.2f}')
print(f'Accuracy: {accuracy:.2f}')
```

```
Mean squared error: 1.13
Root Mean squared error: 1.06
Mean absolute error: 0.80
R2 score: 0.52
Accuracy: 0.55
```

We obtain a RMSE of 1.06 and now let’s see the matrice of confusion



We observe that the model has some difficulties to predict the middle rating (2-3-4), but it has a good comprehension of the extreme rating like 1 or 5.

For example, it has predicted 640 true rating 1, but the modele predicted 1068 rating as 2, and there are actually rating 1.

It is the same idea for the other extreme rating (number 5) : it has predicted 188 true rating 5, but it as also predicted 995 as rating number 4, but they are actually rating number 5.

Also when it is real rating 1 or 2, the modele predict with big error (rating 4 or 5) less than 100 reviews (on total of more than 3K reviews). We observe the same tendency for the review 5.

This model predict very well the difference between “satisfied” reviews and “not satisfied” reviews.

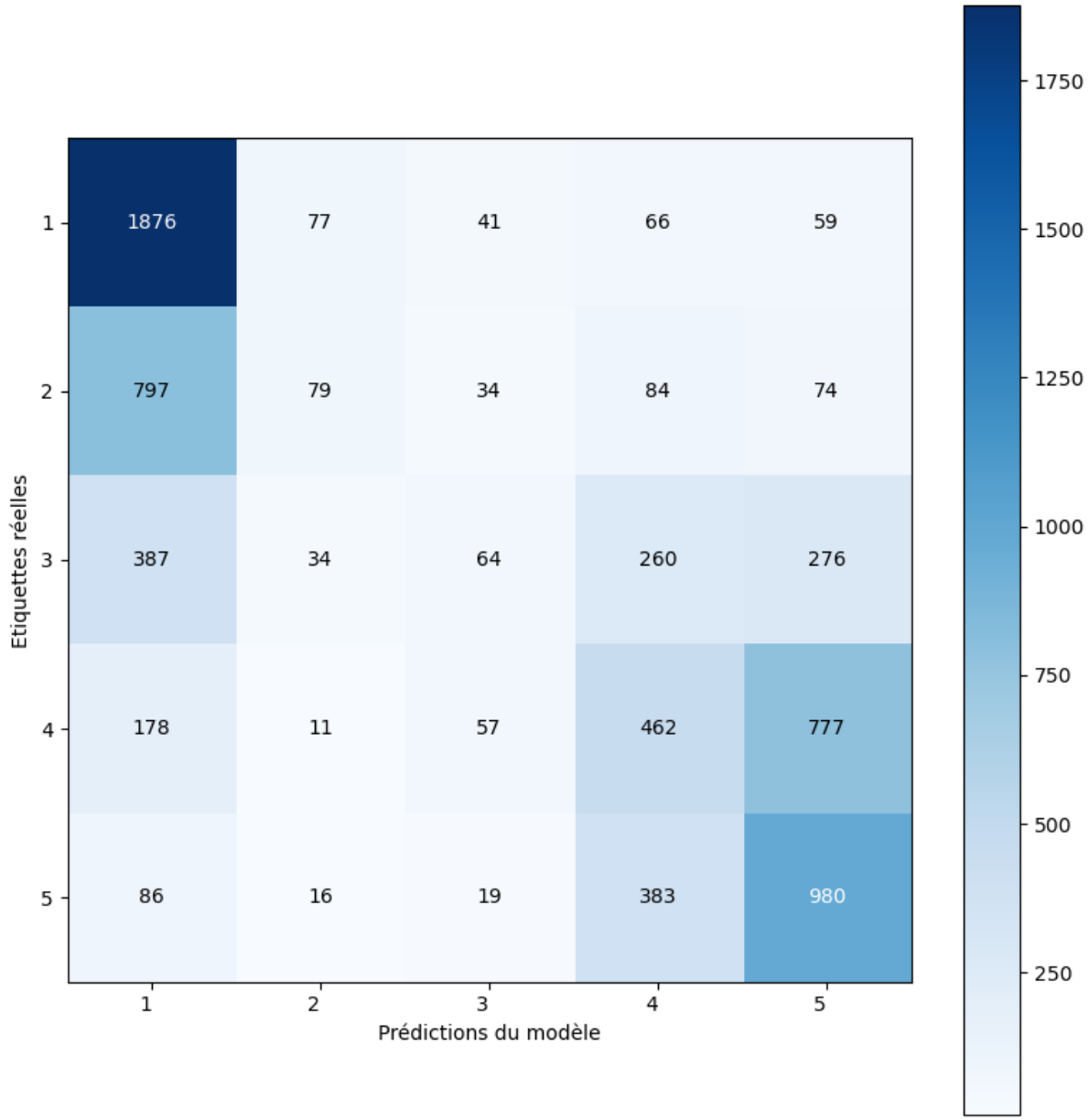
- **Logistic Regression**

```
MSE = mean_squared_error(Y_test, y_pred)
RMSE = math.sqrt(MSE)
mae = mean_absolute_error(Y_test, y_pred)
r2 = r2_score(Y_test, y_pred)
accuracy = model.score(X_test,Y_test)

print(f'Mean squared error: {MSE:.2f}')
print(f'Root Mean squared error: {RMSE:.2f}')
print(f'Mean absolute error: {mae:.2f}')
print(f'R2 score: {r2:.2f}')
print(f'Accuracy: {accuracy:.2f}')
```

```
Mean squared error: 1.54
Root Mean squared error: 1.24
Mean absolute error: 0.79
R2 score: 0.35
Accuracy: 0.48
```

We obtain a RMSE of 1.24 and now let’s see the matrice of confusion



Same as the previous model, but here, the model doesn't understand the middle rating like 2-3-4.

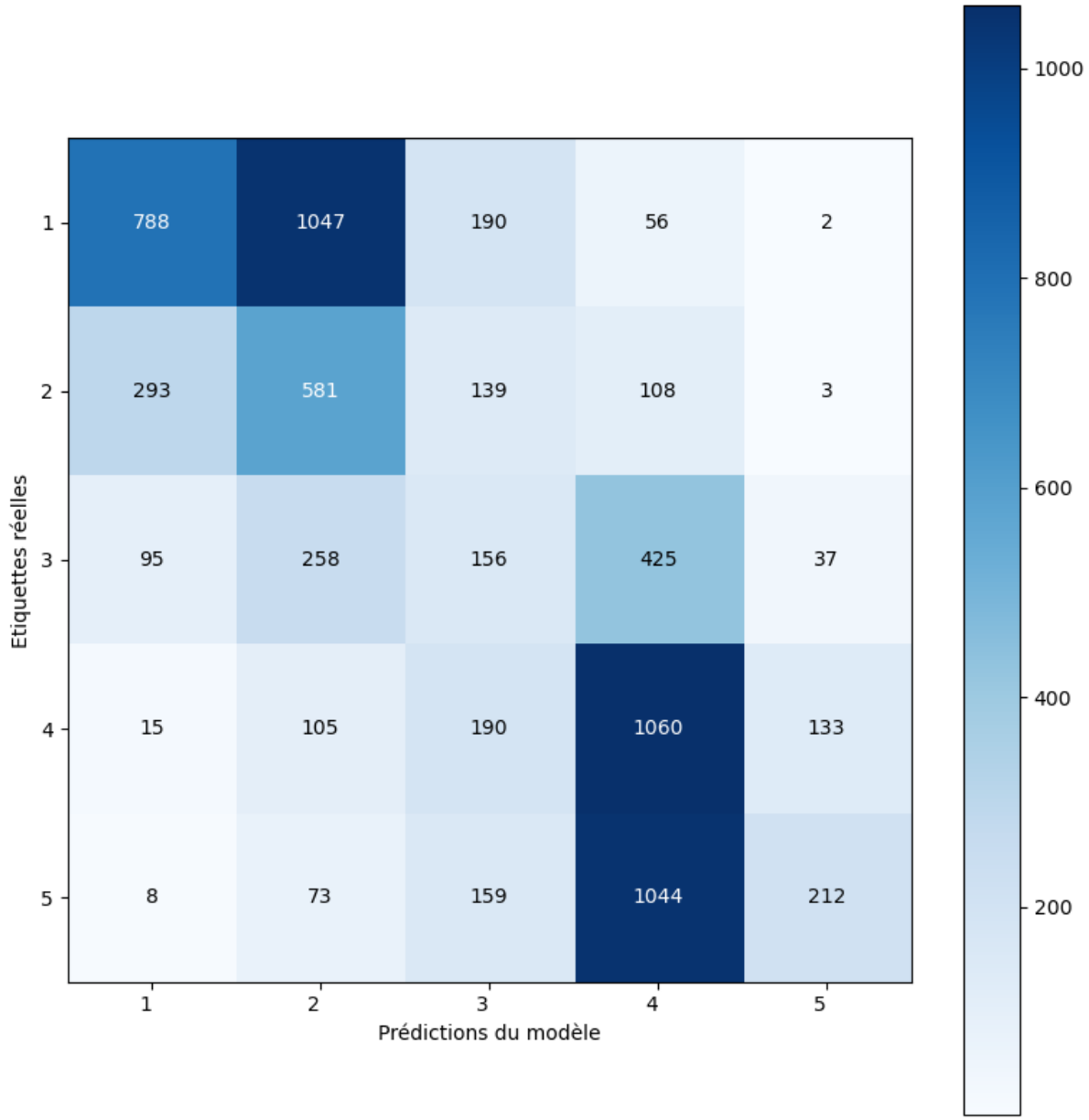
The previous tendency observed is not as good. We are not going to keep this model.

- XGB Regressor

```python
MSE = mean_squared_error(Y_test, y_pred)
RMSE = math.sqrt(MSE)
mae = mean_absolute_error(Y_test, y_pred)
r2 = r2_score(Y_test, y_pred)

print(f'Mean squared error: {MSE:.2f}')
print(f'Root Mean squared error: {RMSE:.2f}')
print(f'Mean absolute error: {mae:.2f}')
print(f'R2 score: {r2:.2f}')
```

```
Mean squared error: 1.09
Root Mean squared error: 1.04
Mean absolute error: 0.75
R2 score: 0.54
```



XGboost has the best RMSE than the two others.

When we observe the matrice of confusion, we see that it predict very well the extreme value like 1 and 5 and it has a better comprehension of the rating 2 and 4,

For example, we have:

- for rating 1 : 788 true rating 1 but 1047 predicted as rating 2, and he understand very well the difference between rating 1 and rating 4 and 5
- for rating 2 : 581 true rating 2, but 293 predicted as rating 1 and 139 as rating 3.
- for rating 3: it has still difficulties to understand
- for rating 4: 1060 true rating 4 : this a very good results

- for rating 5 : 212 true rating 5, but 1044 predicted as rating 4.

Here the model has difficulties to understand the difference between 1-2 and 4-5, but it gives us a first good model that we can work on it, in order to have a better score of precision.

We observe the same tendency as the model linear Regresssor.

IV. Conclusion and self-criticism:

In order to have optimize and have better model, we could add some modification on our project:

- optimize our models with librairies like hyperopt (we used it but without good results).
- by creating new feature, such as the difference of the date of reviews and the date of the event or the global sentiment of the “avis”.
- try other rate for the presence of some word and then compile again the model.
- Maybe it could be interesting to not reduce the dimensionality in the pca.

Despite all of this, in the end, it remains to rethink our overall reasoning and its default. Indeed, we tried to find the optimal solution for selecting "interesting" words, but in the end, for our vector, we filtered based only on the sentimental aspect of words. A combination of their rate of presence in the different classes and their total number of appearances may be a good lead to explore. We tried to combine them without success (column avis3), but we remain convinced of its potential.

In addition, filtering words by their sentiment can be very questionable, especially when taking a word out of its overall context. We did some tests to see if the complete sentiment of the review differed greatly from our "sentiment" value calculated from the retained words, but without going any further.

Final Interpretation : Why the rating is low ?

The words used in the reviews are really specific for a certain rating. As we saw with the word-cloud in the rating 5, the words ‘satisfait’ or ‘recommand’ are present in the

rating 4 too. It is in this case very difficult for a model to distinguish these two rating.

On the other hand, we can find words like ‘fuir’ or ‘sinistre’ in the rating 1. In general, there is no positive words. That’s why it is very easy for a model to distinguish the rating 1 from the rating 5. Using the sentiment of a word, we are able to see the differents between “satisfied” and “not satisfied”. That’s exactly what we saw in the confusion matrix. To conclude, the negative words in reviews have a major impact in the low rating and the positive words have a major impact in high rating.

Finally, despite everything, we are proud of our work, of the data analysis and preprocessing. We find that our WordToVec is relevant and that the addition of the sentimental value for the unsupervised is coherent and allows for a better model. Similarly, for the supervised, we are satisfied with our new features and, even if the score can be greatly improved, our confusion matrices seem to indicate that the separation between positive and negative ratings is well predicted.

Note about Prediction

We used XGBoost to make the predictions. By doing the same preprocessing as the one we did for the train data we remove 1000 lines over the 10k. Obviously we know that it is a mistake (we remove lines by “assureur”) but we did not have the time to fix it.