Promotion 2022
5^e année,

ESILV
Data et intelligence artificielle
France/Courbevoie

Natural Language Processing: Final Project

Nicolas Kevin
Alizadeh Asle Saisan Charly Kyan

1 Data

1.1 Cleaning

The first column to clean was the `date` column that we converted into `Datetime` object. All the dates had the same format, for example the first row had the following string for the `date` column: *"06 septembre 2021 suite à une expérience en septembre 2021"*. To convert it into `Datetime` object we first replace the French months by their respective two-digit format (`janvier: "01", fevrier: "02", etc`), then we strip the beginning trailing whitespaces and only keep the first 10 characters. Then we use the `pandas`' function `to_datetime()` to convert the column.

We also subtracted 1 to all the `note` column so that the target starts at 0.

1.2 Exploration

First, we looked into the stars' distribution for all the insurer (fig. 1).

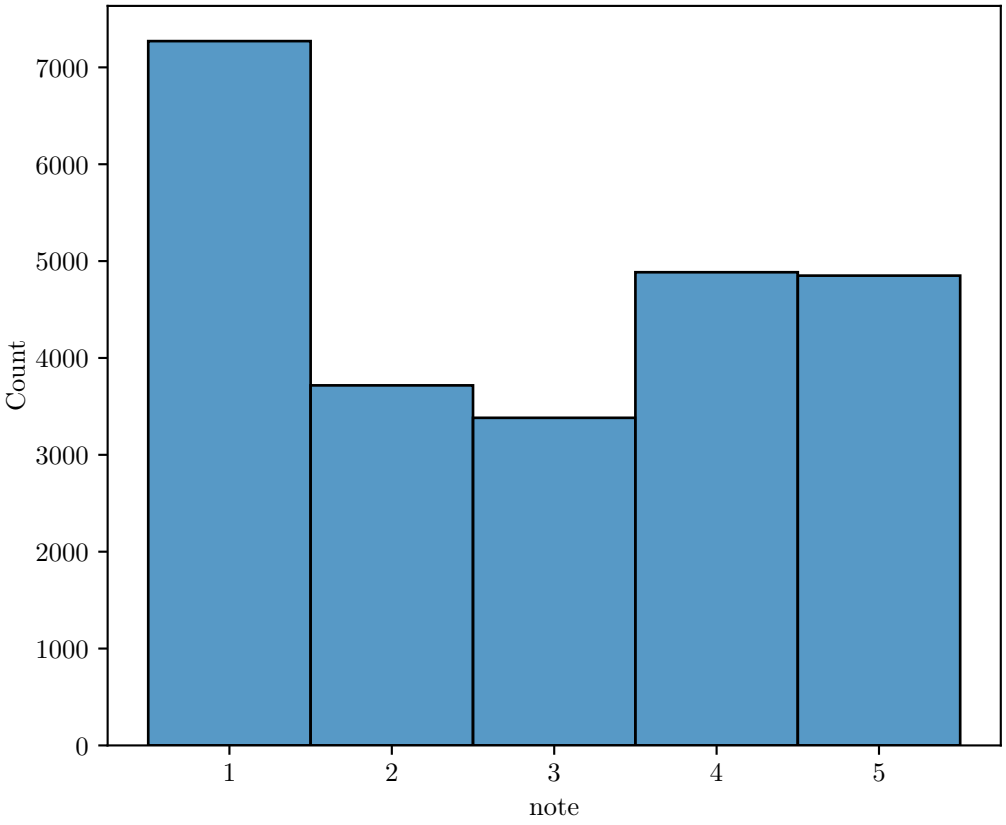

Figure 1: Stars' distribution

Then we looked into the stars' distribution per insurer without and without scaling the y axis (fig. 2 and fig. 3).

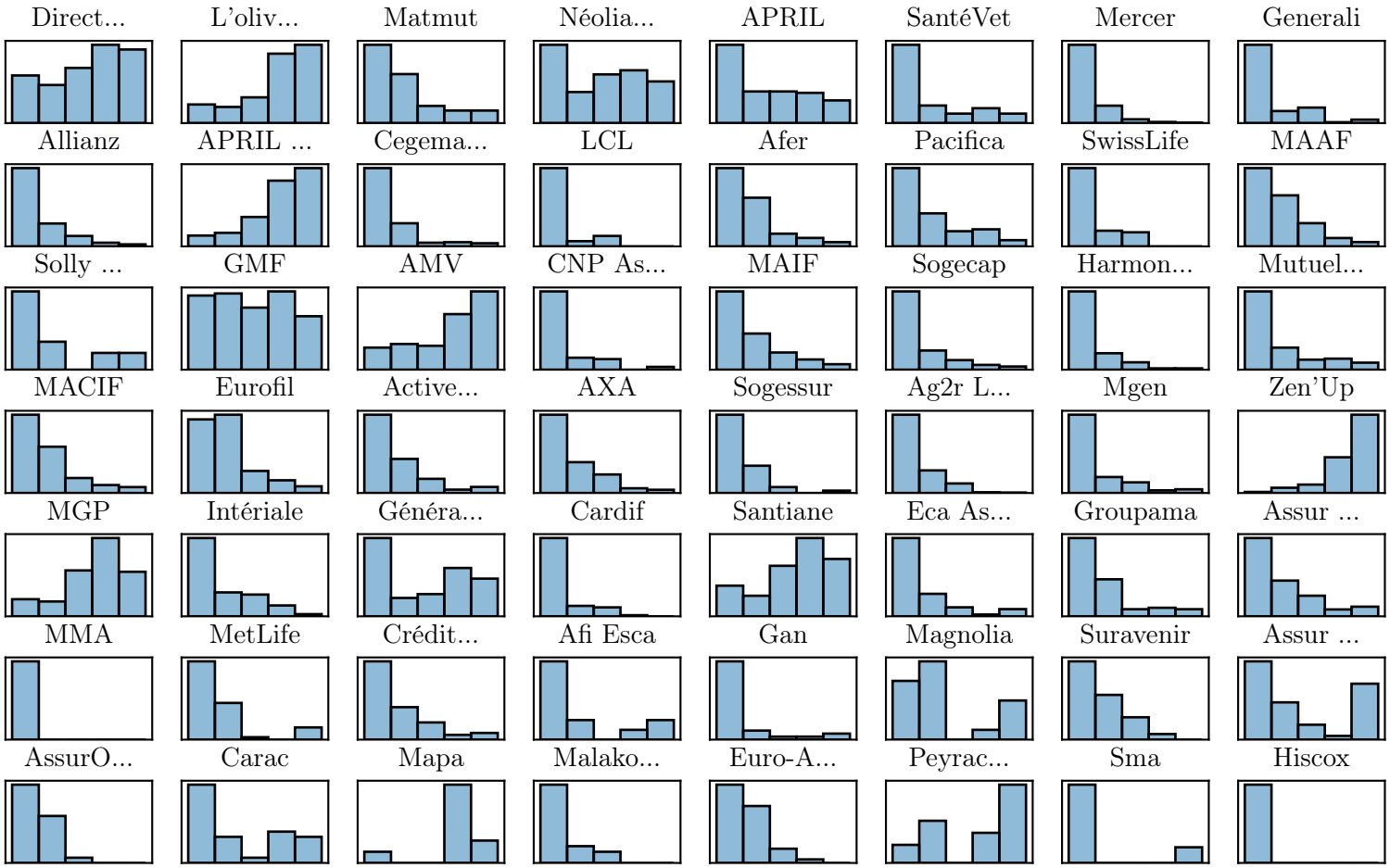

Figure 2: Stars distribution per insurer

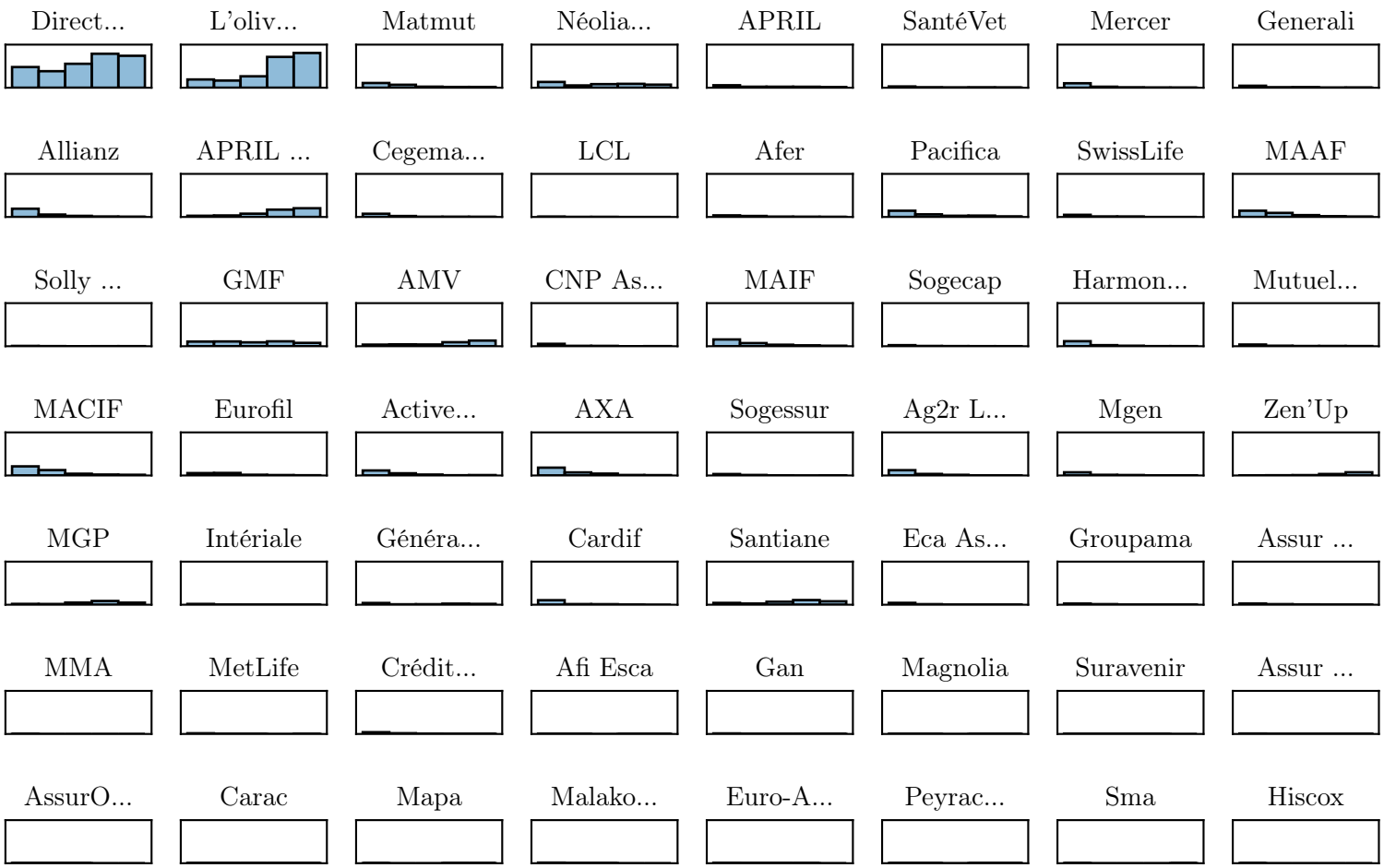

Figure 3: Stars distribution per insurer (y-axis scaled)

We also looked at the mean note per insurer, we used a gradient color to show the number of ratings per insurer, the gradient intensity is defined by the number of ratings in fig. 4 and by the rank of the insurer (ordered by number of rating) in fig. 5

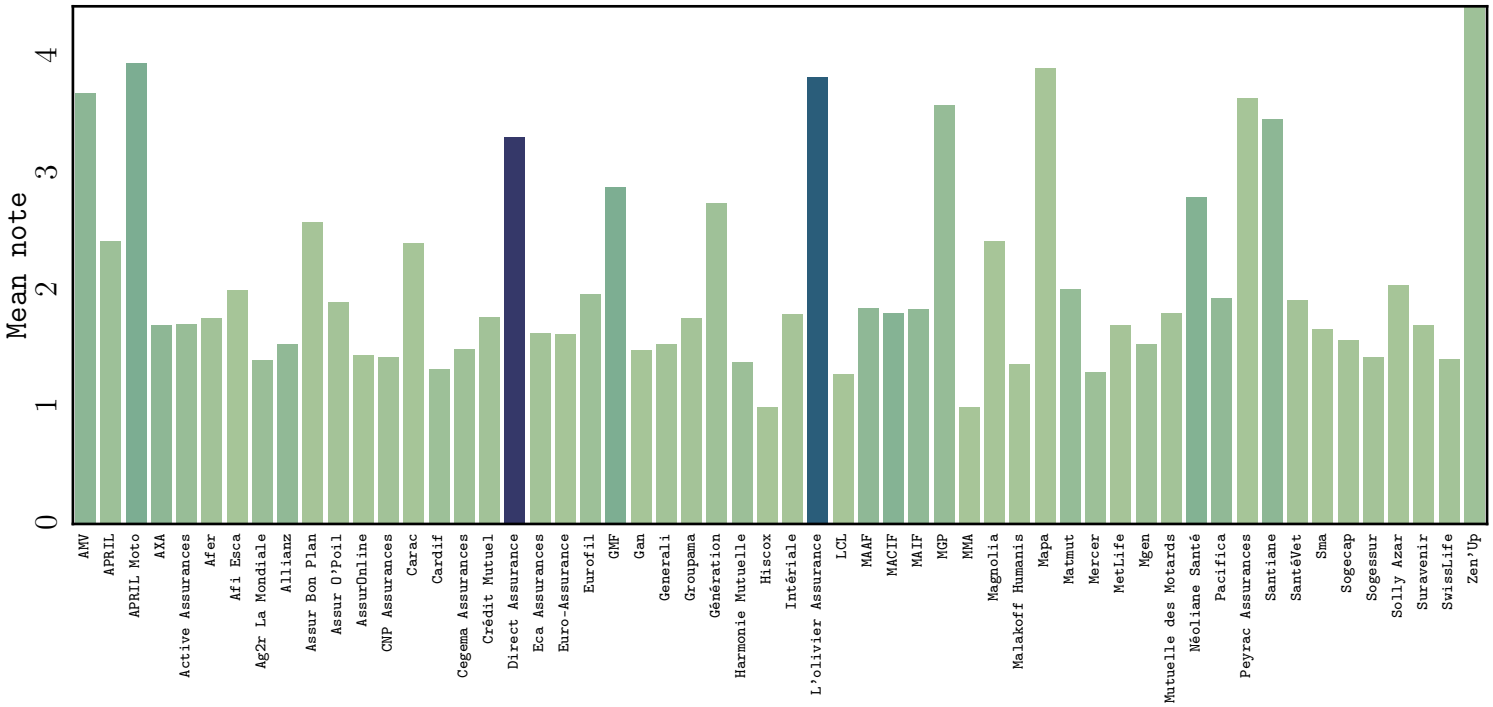

Figure 4: Mean note per insurer (colored by number of ratings)

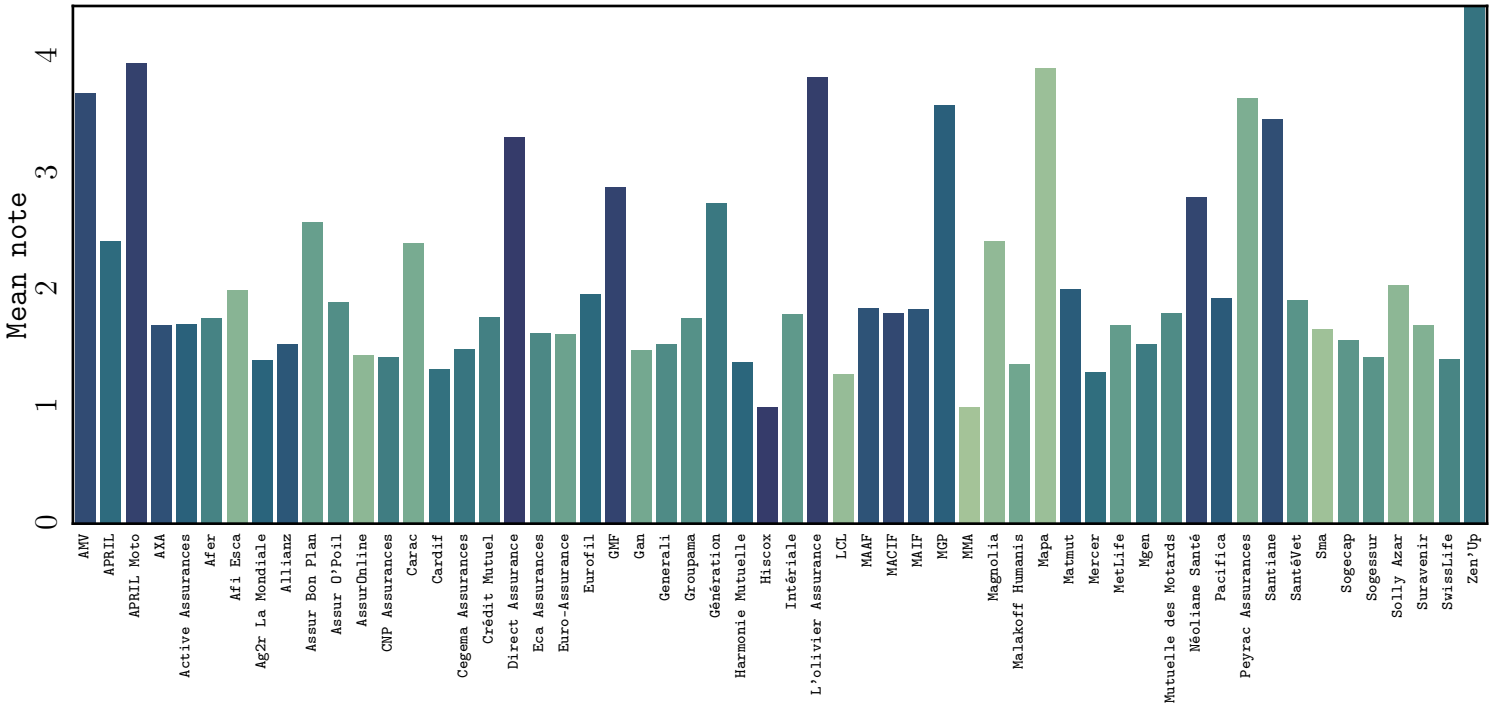

Figure 5: Mean note per assureur (colored by rank)

And so naturally we looked which insurers were the most represented in the train dataset (fig. 6).

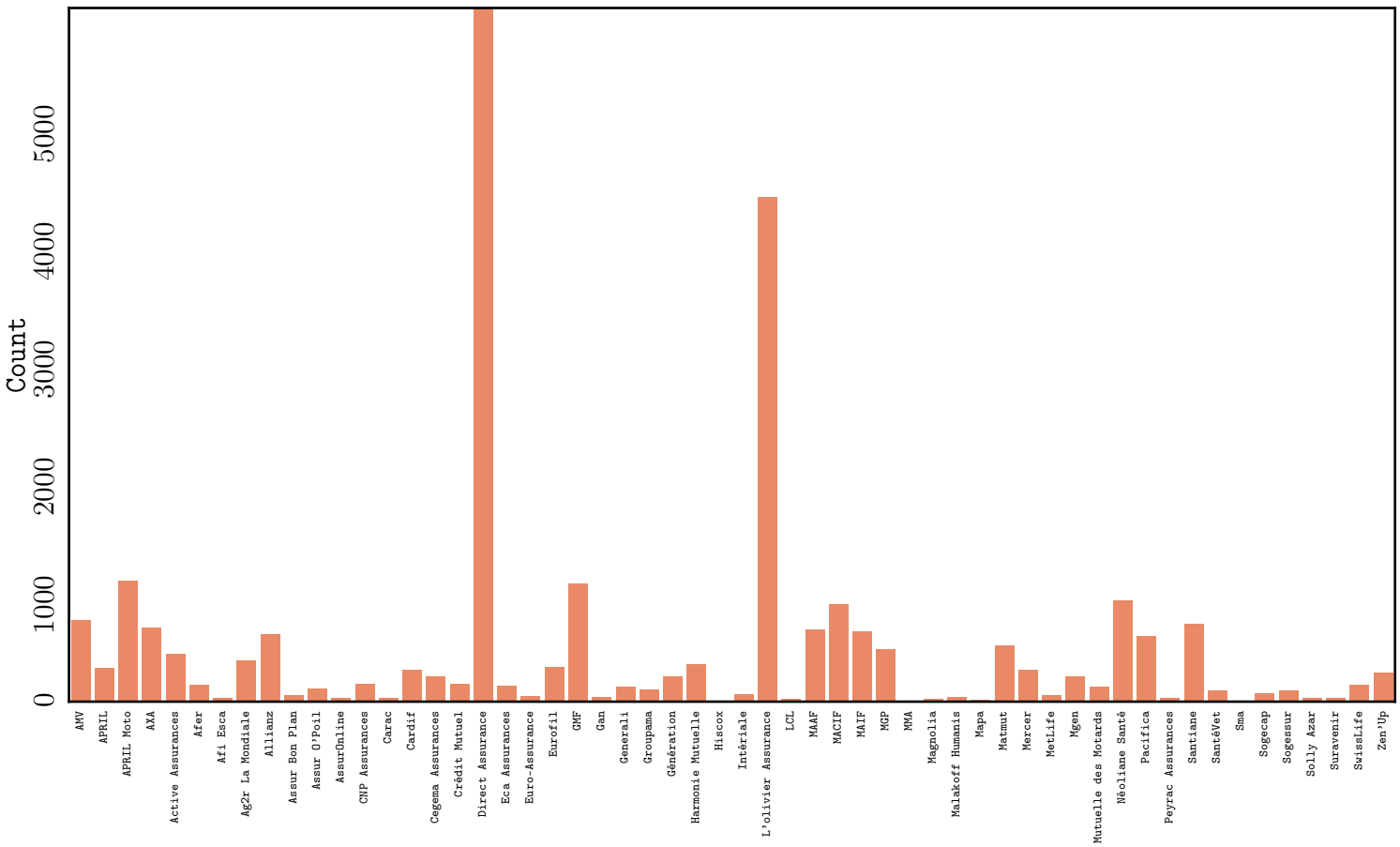

Figure 6: Number of notes per insurer

Finally we watch the number of reviews per date using a calendar

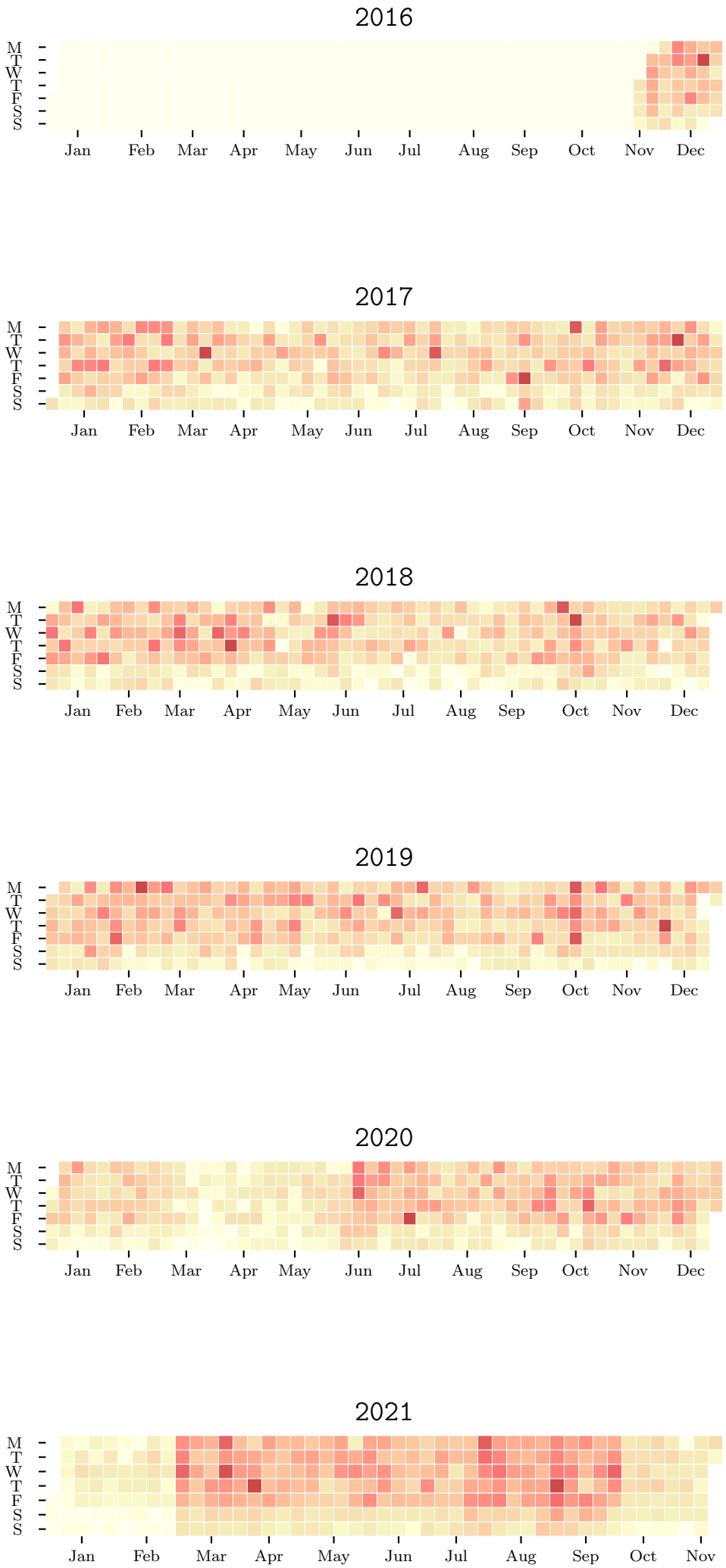

Figure 7: Number of reviews per day

2 Unsupervised

We chose to use the Latent Dirichlet Allocation model to extract the main topics of the `avis` column. We trained 6 models in totals, one on all the `avis` concatenated, and one for each `avis` concatenated by their respective `note`.

2.1 Data processing

We apply the following processing for the LDA models:

- Word-tokenize the `avis` using `spacy`
- Remove punctuation, stop words and words under 3 characters
- Strip and lower the words
- Lemmatize
- Stemmatize

This processing applied to the first `avis` gives us the following results:

Original	**Processed**
Meilleurs assurances, prix, solutions, écoute, rapidité, et je recommande cette compagnie pour vous \r\nDes prix attractif et services de qualité et rapidité	meilleur assur prix solut écout rapid recommand compagn prix attract servic qualit rapid

2.2 Model

The section 2.2 shows the result for the LDA model trained on all the `avis` and the section 2.2 shows the results for the LDA models trained on the `avis` grouped by their `note`.

Topic	**Words**
0	[assur, servic, prix, contrat, être, bien, client, demand, fair, mois]

Table 1: LDA result

Nothing too surprising with those results, the stemmatized words we get are all pertinent in the context of assurance review.

Note	**Words**
0	[assur, contrat, mois, **aucun**, être, demand, fair, **rembours**, pai, client]
2	[assur, être, contrat, **sinistr**, demand, mois, **aucun**, fair, servic, véhicul]
3	[assur, servic, prix, contrat, **bien**, être, **satisf**, fair, client, **rapid**]
4	[assur, prix, servic, **satisf**, **rapid**, **bon**, **bien**, conseil, être, tarif]
5	[assur, prix, servic, **satisf**, **rapid**, **recommand**, **bon**, **bien**, conseil, tarif]

Table 2: LDA results on grouped reviews (we highlighted the word that we felt were the most pertinent)

The results seem to be coherent the note given for the review, with bad connotated word for the note 1 and 2 (*rembours, aucun*) and positively connotated words for the note 3, 4 and 5 (*bien, satisf, bon, rapid, recommand*).

We could maybe improve the pertinence of those results with more preprocessing on the data (maybe remove some very generic verb such as *être* and *faire*).

3 Supervised

For the supervised task we use word embedding to vectorize the `avis` column, then we use this embedding in addition to other features to train multiple models.

3.1 Data processing

We use `spacy` and `fasttext` word embedding to vectorize the column `avis`, both give 300 size long vectors. We converted the columns `assureur` and `produit` into categorical variables. Finally we extracted the `dayofweek, day, month` and `year` from the from the `date` column, then we dropped the `auteur` and `date` columns.

3.2 Models

3.2.1 Random Forest

For the Random Forest model we used the sklearn implementation without depth limit.

True\Predicted	1	2	3	4	5
1	5818	0	0	0	0
2	0	2999	0	0	0
3	0	0	2745	0	0
4	0	0	0	3860	0
5	0	0	0	0	3861

Table 3: Train confusion matrix

We obviously attained an accuracy of 1 and a RMSE of 0 for the training set.

True\Predicted	1	2	3	4	5
1	1319	38	23	47	26
2	592	26	19	52	28
3	285	31	33	178	110
4	168	11	50	412	384
5	99	8	31	318	533

Table 4: Val confusion matrix

Accuracy	0.48185023853972203
RMSE	1.642190416925949

Table 5: Val metrics

3.2.2 Feed Forward Neural Network

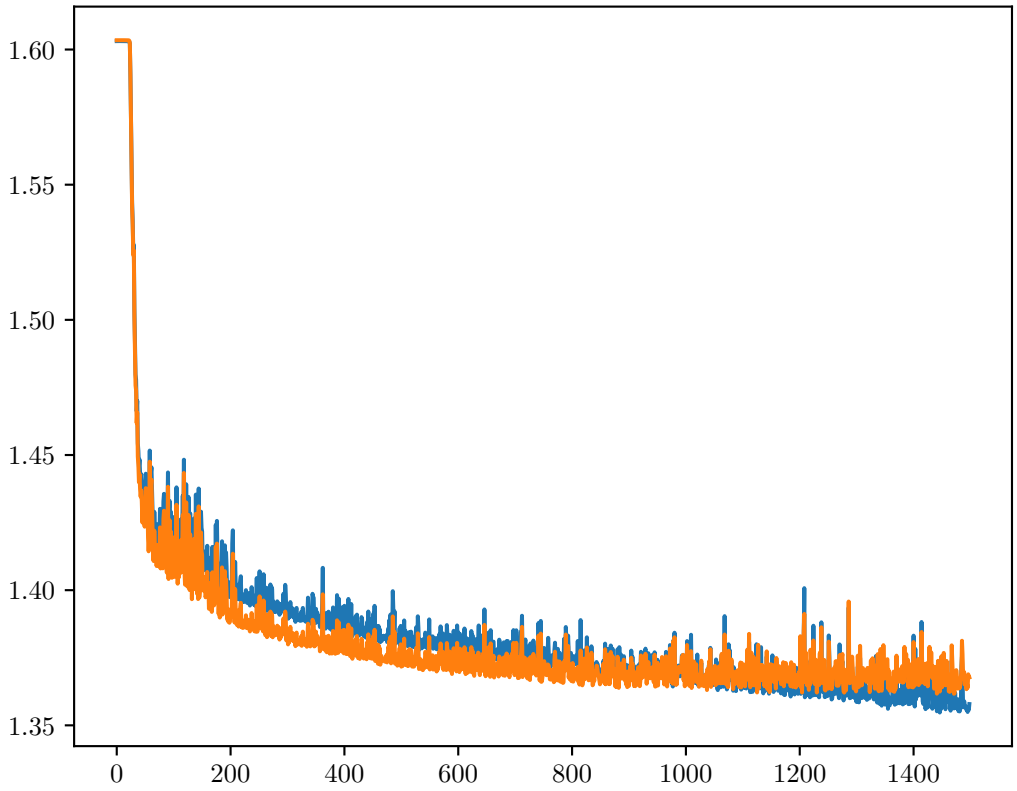

Figure 8: Loss train

Accuracy	0.5394388839910802
RMSE	1.217445418244049

Table 6: Train metrics

True\Predicted	1	2	3	4	5
1	5592	27	51	79	69
2	2619	64	92	142	82
3	1180	90	239	791	445
4	363	54	162	1956	1325
5	229	16	63	1002	2551

Table 7: Train confusion matrix

Accuracy	0.5289359054138145
RMSE	1.3432897739058287

Table 8: Val metrics

True\Predicted	1	2	3	4	5
1	1384	11	17	24	17
2	606	16	22	50	23
3	276	13	49	188	111
4	121	11	54	472	367
5	75	4	18	263	629

Table 9: Val confusion matrix

We could achieve better than a RMSE of 1.34 on the validation dataset using feed forward neural network.

We achieved these results using the following architecture:

```
Linear(306, 256)
ReLU()
Linear(256, 128)
ReLU()
Dropout(0.5)
Linear(128, 128)
ReLU()
Dropout(0.5)
Linear(128, 5)
Softmax()
```

3.2.3 CamemBERT

We tried to use the CamemBERT model but neither our GPU or Google Colab's GPU could handle more than a `batch_size` of 1 for the training, which were taking too much time to train. You still can find the code for this experiment in the repository.

Also we didn't feel confident with the code for this experiment, so it is most likely `that the out of memory` errors are the result of our lack of understanding.

4 Code

You can find all the code on github: `https://github.com/charlyalizadeh/ESILV_NLP`