Le but de l'exercice de prédiction

On se met à la place d'un responsable de Marketing/Ventes ou de la Supply Chain d'une compagnie que commercialise une certaine (inconnue) variété de produits et services. On a reçu un ensemble de données qui correspondent aux notations et « reviews » (commentaires explicites sur les différents aspects des produits ou services : logistique, délai de livraison, qualité, prix, service après-vente, …) de 12000 utilisateurs de nos produits et services.

On voudrait s'en servir de ces informations pour déterminer si on est capable de prédire la notation (« rating ») qu'un client quelconque va donner à notre compagnie lorsqu'on reçoit un « review » de ce client. En principe ce dernier peut choisir une et seulement une notation parmi les valeurs 1, 2, 3,4, 5.

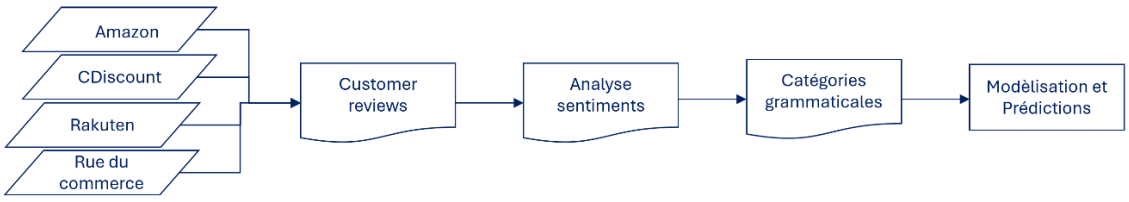
Evidemment on peut aussi penser qu'il pourrait être intéressant de prédire, à partir d'un « review », si le client que l'a rédigé est un détracteur de notre compagnie, une personne neutre ou quelqu'un qui est plutôt un promoteur de la même. Par ailleurs, cette approche permettra de faire une segmentation de nos clients qui ont acheté nos biens ou services, ce qui pourrait signifier qu'on serait capable d'appliquer un politique commercial et une communication différente envers chaque segment. Cette option est donc plus intéressante, pour ses applications, que simplement prédire le « rating ».

Cependant on peut définir las catégories « détracteur », « neutre » et « promoteur » à partir de la variable « rating ». On pourrait faire : « détracteur » si le « rating » est égal à 1 ou à 2, « neutre » si le « rating » est égal à 3, et « promoteur » si le « rating » est égale à 4 ou 5. Una autre avantage de cette approche est qu'on groupe les populations selon le « rating » en réduisant ainsi les nuisances, dans l'estimation des modèles et dans la prédiction, dues à la présence de populations de taille trop inégales. Dans notre travail d'estimation et de prédictions nous explorons ces idées et présentons les résultats.

On commence avec une présentation des données, leur traitement ou « pre-processing », enfin on présente les différents modèles estimés et les prédictions réalisées.

1. Les données d'entré

L'obtention des données et leur traitement est décrit dans le schéma suivant :



En utilisant le « web scrapping » on a obtenu 12000 avis des consommateurs de ces plateformes, environ 3000 observations par plateforme. Après nettoyage des données, on a appliqué l'analyse de sentiments d'une part et ensuite on a extrait le nombre de mots utilisées et le nombre des différentes catégories utilisées dans les « reviews » : sujets, verbes, adverbes, exclamations, … L'extraction des catégories grammaticales a été effectué en utilisant la librairie **spacy**.

Les variables retenues sont :

plate_forme: chacune des plateformes de e-commerce
r**ating**: notation atribuée par le client à chaque plateforme
nb_tokens: longueur du text écrit par le client, nombre de mots utilisée dans le "review"
category_numeric: -1 (détracteur), 0 (neutre), +1 (promoteur)
adjs_counts: nombre d'adjectifs dans le "review"
nouns_counts: nombre de noms dans le "review"
pronouns_counts: nombre de pronoms dans le "review"
prop_nouns_counts : nombre de pronoms propres dans le "review"
adverbs_counts : nombre de adverbes dans le "review"
verbs_counts : nombre de verbes dans le "review"
interjections_counts : nombre de interjections dans le "review"
symbols_counts:nombre de symbols dans le "review" (?, !, ...)

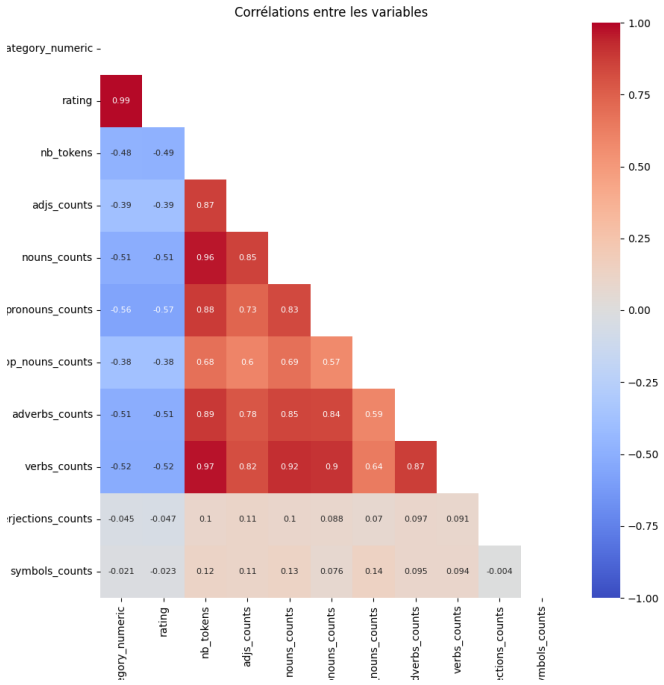
Une description simple des variables est montrée ci-dessous :

Item	plate_forme	rating	nb_tokens	category_numeri	adjs_counts	nouns_counts	pronouns_count	prop_nouns_cou	adverbs_count	verbs_counts	nterjections_cou	symbols_count
count	12 650	12 650	12 650	12 650	12 650	12 650	12 650	12 650	12 650	12 650	12 650	12 650
mean	1,49	2,87	63,45	-0,05	3,57	11,01	4,41	1,53	4,30	7,72	0,00	0,00
std	1,12	1,89	84,33	0,99	4,09	12,65	5,18	2,17	4,65	10,07	0,06	0,07
min	0	1	1	-1	0	0	0	0	0	0	0	0
25%	0	1	10	-1	1	2	0	0	1	1	0	0
50%	1	2	36	-1	2	7	3	1	3	4	0	0
75%	2	5	84,75	1	5	15	7	2	6	11	0	0
max	3	5	1371	1	55	220	38	67	44	119	1	1

On voit que la variable « symbols_counts » n'apporte pas d'information relevant.

2. Les corrélations entre les variables et les caractéristiques des sous-populations

Les corrélations (de Pearson) entre les variables d'interet sont montrées à continuation :



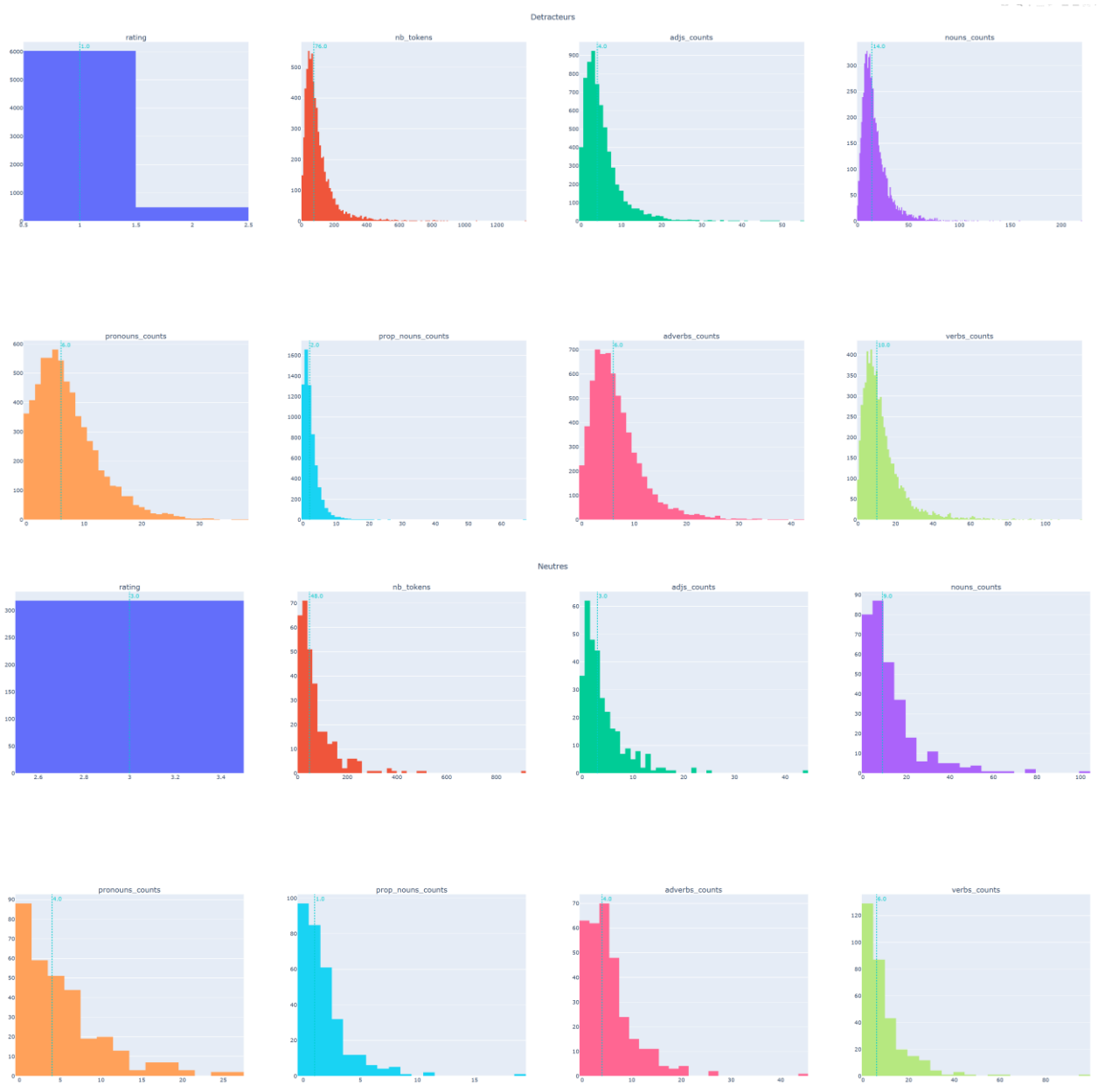
Vu que la définition de la variable «catégory_numeric » (détracteurs, neutres, promoteurs) a été faite à partir du « rating », la forte corrélation positive entre ces deux variables n'est pas une surprise.

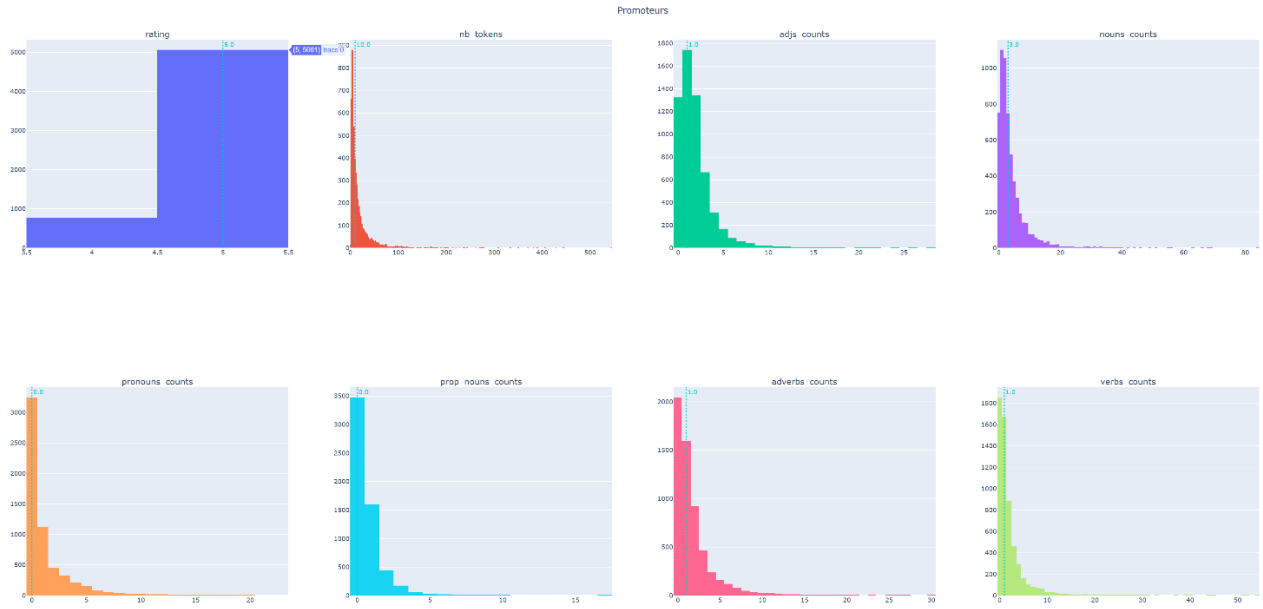
La forte corrélation entre « nb_tokens » et le nombre des différentes catégories grammaticales utilisées n'est pas surprenant non plus : plus long est le texte, plus de catégories grammaticales utilisées. C'est juste une relation mécanique.

Par ailleurs, l'utilisation dans un texte des verbes, des adverbes, des pronoms, etc varie positivement. Les catégories grammaticales tendent à s'utiliser ensemble : plus de verbes et des adverbes généralement impliquent plus de pronoms et de pronoms propres.

On est donc face à la présence d'un problème de multicolinéarité entre les variables explicatives ou «features » retenues. C'est un élément à prendre en compte dans la spécification de nos modèles d'estimation.

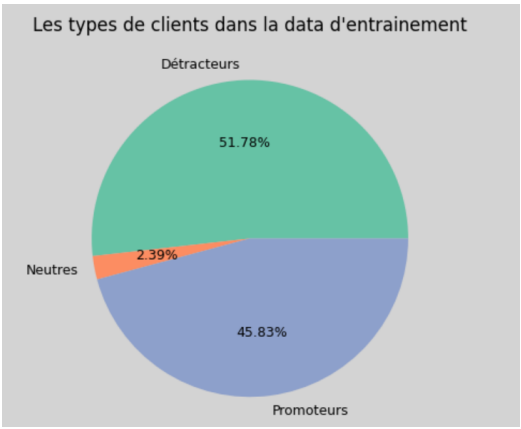
En ce qui concerne à la statistique descriptive des « features » selon la catégorie de clients (détracteurs, neutres, promoteurs) on a :





On observe que les distributions empiriques des variables ne sont pas symétriques et que la longueur du texte, et donc aussi le nombre des fois que chaque catégorie grammaticale est utilisée, varient selon avec le segment auquel le client appartient : **pour un client détracteur** le **nombre de fois que chaque catégorie est utilisée est plus grande** (quel que soit la catégorie) **que pour un client promoteur**. Le cas le plus représentative (et n'est pas le seul) est l'utilisation des adverbes. *«Les adverbes sont des mots invariables qui se joignent à des verbes, des adjectifs, des prépositions, des phrases ou d'autres adverbes pour en modifier ou en préciser le sens »* (https://dictionnaire.lerobert.com/guide/qu-est-ce-qu-un-adverbe). Ils apportent donc directement des informations sur l'état d'esprit du client.

On doit signaler que les données par segment de client sont déséquilibrées :



En fin, il faut signaler qu'avant de travailler la modélisation on a normalisé les variables et lorsque on a utilisé des modèles de « deep learning » on aussi applique « OneHotEncoder » à la variable cible.

3. Les modèles travaillés et leurs prédictions

Etant donné un ensemble d'observations notre but c'est de prédire, à partir des « features » disponibles, si le client est un détracteur, c'est neutre ou plutôt un promoteur de notre

compagnie. La variable « target » peut prendre une seulement une valeur possible : détracteur, neutre ou promoteur ; après encodage de la variable cible on aura, respectivement : -1,0, ou +1. Il s'agit d'un problème de prédiction multi classe (trois classes possibles) avec des classes déséquilibrées.

Dans l'estimation des modèles que sont présentés ci-dessous, on a commencé par séparer un sous-ensemble de training et un autre de test et ensuite on a normalisé les variables pour enlever « l'effet échelle »

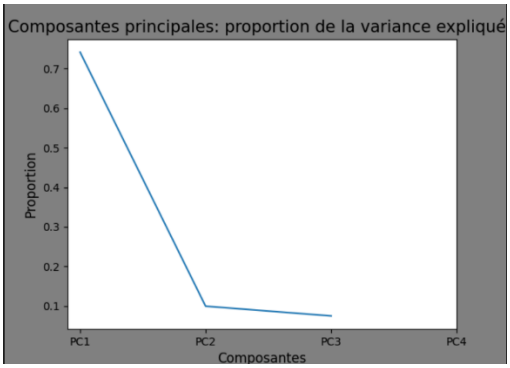
```
# définition target/features:
target = df['category_numeric'].copy()
# on enleve les variables qui ont une corrélation très faible avec le target:
feats = df.drop( labels: ['category_numeric', 'interjections_counts', 'symbols_counts', 'nb_tokens', 'plate_forme'], axis=1).copy()
# split trian/test
X_train, X_test, y_train, y_test = train_test_split( *arrays: feats, target, test_size=0.25, random_state=42)

# scaling la data
scaler_train = StandardScaler()
scaler_test = StandardScaler()

X_train = scaler_train.fit_transform(X_train)
X_test = scaler_test.fit_transform(X_test)
```

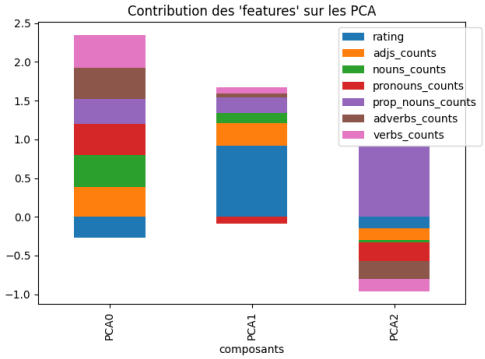
a) Le modèle de régression logistique multinomial

A partir de ScikitLearn on a défini le modèle (avec « cross validation) :

```
# Définition du modèle de regréssion logistique multinomiale (avec cross validation)
model_lg = LogisticRegressionCV(cv=10, random_state=0).fit(X_train_pca, y_train)
```

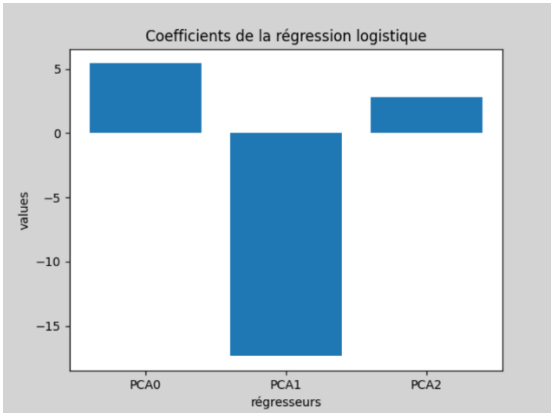
Pour diminuer les effets de la multicolinéarité, signalés auparavant, on a utilisé l'analyse de composants principales (PCA en anglais), le but c'est de réduire le nombre de variables et, surtout, rester avec un ensemble de nouvelles « features » (les PCA) qui seront orthogonaux entre eux. On a choisi de sélectionner les PCA qui expliquent 90% de la variance, ce qui nous a rendu trois composants : PCA0, PCA1 et PCA2.





On apprécie que le premier composant explique plus de 75% de la variance et que dans ce composant (PCA0), toutes les « features » contribuent presque dans la même mesure. Le composant PCA1 explique presque 10% de la variance et c'est le rating que joue un rôle important dans sa définition. Le dernière composante, PCA2, explique un peut moins de 5% et c'est le nombre d'adverbes la variable que contribue le plus à sa définition.

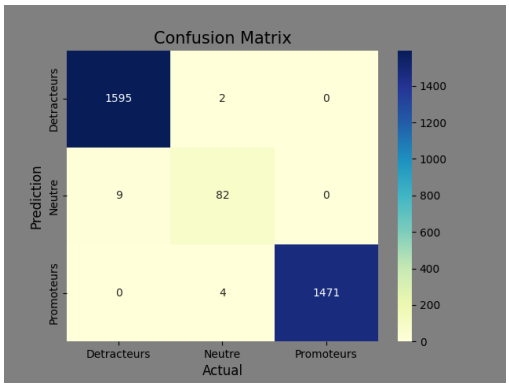
En ce qui concerne aux résultats de l'estimation, on a :




```
Accuracy su modèle
score sur le training data:  0.9966269632128175
score sur le testing data:  0.9952576667720519
```

Et on n'a pas trouvé d'évidence de « surestimation » et les prédictions sur le sous-ensemble de test sont :

```
Classification report
              precision    recall  f1-score   support

          -1       0.99      1.00      1.00      1597
           0       0.93      0.90      0.92        91
           1       1.00      1.00      1.00      1475

    accuracy                           1.00      3163
   macro avg       0.98      0.97      0.97      3163
weighted avg       1.00      1.00      1.00      3163
```



On peut dire, alors que les résultats du modèle de régression logistique multinomiale sont globalement bons.

b) Le modèle « deep learning » (Keras), target: la catégorie de clients

On utilise un modèle de couches séquentielles définit avec l'aide de la librairie Keras. On prend en compte le déséquilibre numérique entre les tailles des différentes classes de la variable « target » :

```
class_weights = class_weight.compute_class_weight( class_weight: 'balanced', classes =np.unique(y_train), y=np.array(y_train))
class_weights = dict(zip(np.unique(y_train), class_weights))
```

```
model = Sequential()
model.add(keras.Input(shape=(input_layer_size,), name="input_layer"))
model.add(Dense(input_layer_size * 2, activation='relu', name="first_layer"))
model.add(Dense(hidden_layer_size * 4, activation='relu', name="second_layer"))
model.add(Dropout(0.2))
model.add(Dense(hidden_layer_size, activation='relu', name="ouput_layer"))
model.add(Dense(Y_train.shape[1], activation='softmax'))
```

On compile et estime le modèle en incluant un « callback » de « EarlyStopping » :

```
model.compile(loss='categorical_crossentropy',  # sparse_categorical_crossentropy
              optimizer='adam',
              metrics=['categorical_accuracy'])  # sparse_categorical_accuracy
model.summary()

BATCH_SIZE = 64
EPOCHS = 20
print("\n\n")
print("**" * 36)
print("BATCH_SIZE:", BATCH_SIZE, "EPOCHS: ", EPOCHS)

early_stopping = callbacks.EarlyStopping(monitor='val_loss',
                                         min_delta=0.005,
                                         patience=10,
                                         mode='min',
                                         restore_best_weights=True)

history = model.fit(X_train, Y_train,
                    epochs=EPOCHS,  # 200
                    batch_size=BATCH_SIZE,
                    validation_split=0.2,
                    class_weight=class_weights,
                    callbacks=early_stopping)
```

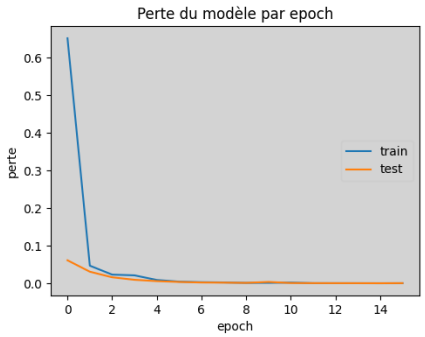
Dans notre contexte, utiliser un modèle de « deep learning » a deux avantages sur un modèle de régression logistique multinomiale. D'abord on prend en compte et on exploite les non-linéarités existantes (soit dans la relation entre les variables, soit dans les paramètres) que par hypothèse on écarte dans le cas du modèle de régression logistique multinomiale ; et en plus on n'a pas besoin d'utiliser des techniques de PCA pour palier le problème de multicolinéarité.

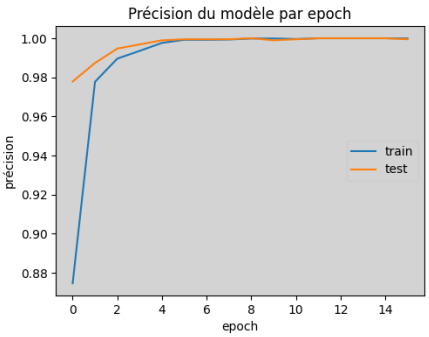
Voici une synthèse du modèle et les résultats sur la qualité des prédictions réalisées sur le sous-ensemble de test :

Model: "sequential"

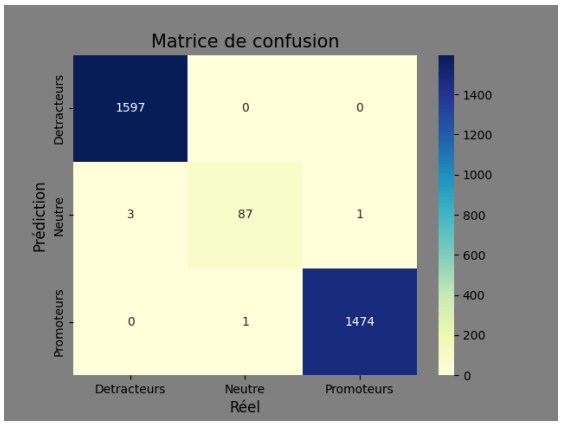
Layer (type)	Output Shape	Param #
first_layer (Dense)	(None, 14)	112
second_layer (Dense)	(None, 448)	6,720
dropout (Dropout)	(None, 448)	0
ouput_layer (Dense)	(None, 112)	50,288
dense (Dense)	(None, 3)	339

Total params: 57,459 (224.45 KB)
Trainable params: 57,459 (224.45 KB)
Non-trainable params: 0 (0.00 B)





	precision	recall	f1-score	support
-1	1.00	1.00	1.00	1597
0	0.99	0.96	0.97	91
1	1.00	1.00	1.00	1475
accuracy			1.00	3163
macro avg	1.00	0.99	0.99	3163
weighted avg	1.00	1.00	1.00	3163

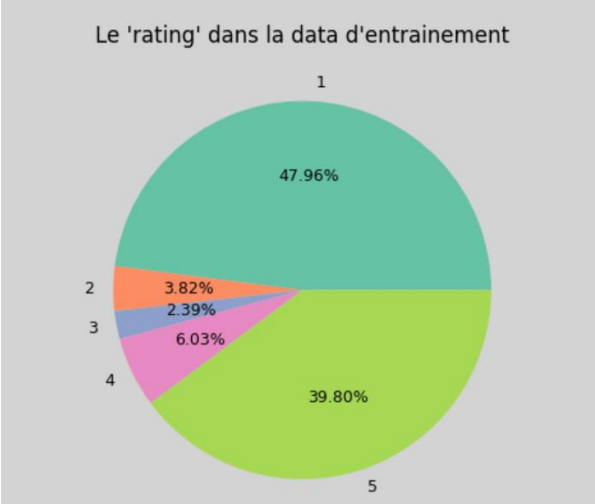


Les résultats ne montrent pas d'indices clairs de « surestimation », les prédictions sont bonnes et leur qualité est meilleure que celle obtenue avec le modèle précédent.

c) Le modèle « deep learning » (Keras), target: le « rating »

Ce modèle montre les difficultés rencontrées pour réussir de prédictions du rating d'une qualité acceptable. On a essayé la régression logistique multinomiale, le XGBoost et l'ADABoost et dans tous ces cas les résultats ne sont pas bons.

La raison de ces difficultés vient du fait qu'il s'agit d'u modèle de prédiction de cinq **classes fortement déséquilibrées** :

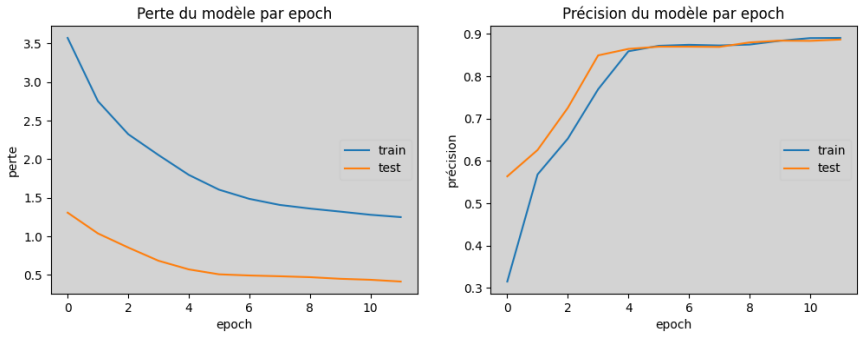


Le modèle que a donné les meilleurs résultats c'est un modèle de couches trouvé après l'avoir entrainé avec différentes combinaisons de tailles de « batchs» et de nombres d'«epochs » :

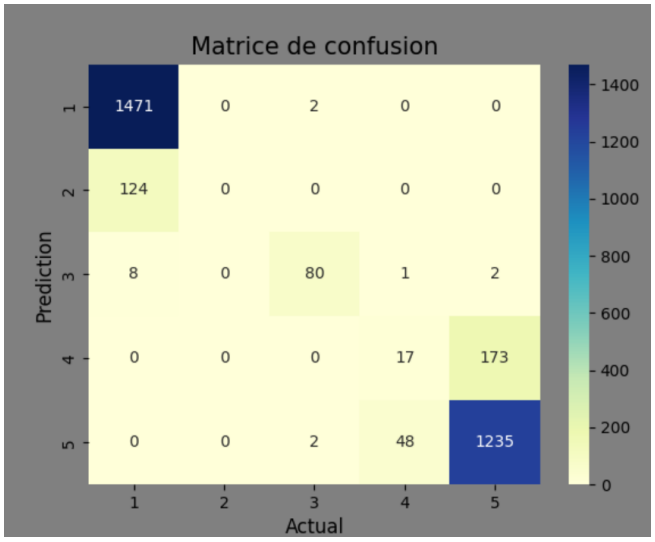
Model: "sequential"

Layer (type)	Output Shape	Param #
first_layer (Dense)	(None, 28)	224
second_layer (Dense)	(None, 56)	1,624
dropout (Dropout)	(None, 56)	0
ouput_layer (Dense)	(None, 224)	12,768
dense (Dense)	(None, 5)	1,125

Total params: 15,741 (61.49 KB)
Trainable params: 15,741 (61.49 KB)
Non-trainable params: 0 (0.00 B)



	precision	recall	f1-score	support
1	0.92	1.00	0.96	1473
2	0.00	0.00	0.00	124
3	0.95	0.88	0.91	91
4	0.26	0.09	0.13	190
5	0.88	0.96	0.92	1285
accuracy			0.89	3163
macro avg	0.60	0.59	0.58	3163
weighted avg	0.83	0.89	0.85	3163



On voit que l'on arrive à bien prédire les classes 1, 3 et 5 mais on a des difficultés à bien prédire les « ratings » 2 et 4.

On a donc deux options : soit chercher un modèle avec une meilleur performance (en incluant peut-être techniques d'échantillonnage), soit grouper les notations. Cette dernière approche, comme signalé auparavant, équivaut à prédire l'attitude du client envers notre compagnie : détracteur, neutre ou promoteur, avec les résultats présentés et discutés dans 3.a) et 3.b).

4. Pour aller plus loin ...

Une possible voie à explorer c'est de continuer à chercher de modèles pour prédire le « rating » en incluant des techniques d'échantillonnage et comparer les résultats obtenus en termes de qualité des estimations et des prédictions et ainsi qu'en termes de temps d'exécution.

Une autre possible piste de travail pourrait être d'augmenter la taille de l'échantillon : si on double la taille de la population, le problème du déséquilibre extrême persiste ? C'est-à-dire, c'est un problème de l'échantillon ou il s'agit d'un problème de la population ?