Projet:

Classification des sentiments sur les critiques d'Allociné

UNIVERSITÉ CÔTE D'AZUR

Groupe:
Alexandre Deprun
Oumaima Khadira

2021 / 2022

Plan

- Prétraitement
- Analyse de données
- Algorithme
- Evaluation du jeu de test
- Conclusion

Première méthode

Prétraitement

- Diviser les données en train et test.
- Créer deux ensemble (critiques positives ,critiques négatives)
- Tokenizer et stop words.

```
from nltk.tokenize.casual import TweetTokenizer
t = TweetTokenizer()
stop_words_fr = stopwords.words('french')
exclude_ponctuation = set(string.punctuation)
stop_words_fr.extend(exclude_ponctuation)
stop_words_fr.extend(["...","’"])
```

```
def tok_sentence(sentence):
  res = []
  res.extend(t.tokenize(sentence.lower()))
  res = [word for word in res if word not in stop_words_fr]
  return res
```

```
def word_feats(sentence):
  tokens = tok_sentence(sentence)
  return dict([(word, True) for word in tokens])
```

```
pos_feats = [(word_feats(review), 'positive') for review in all_positive]
neg_feats = [(word_feats(review), 'negative') for review in all_negative]
```

Première méthode

Algorithme

Naïve Bayes Classifier.

Première méthode

Evaluation du jeu de test

Naïve Bayes Classifier.

accuracy

```
1 accuracy
```

```
0.8946
```

```
1 print(classifier.classify(word_feats('le film est très intéressant')))
```

```
positive
```

Deuxième méthode

Prétraitement

- Diviser les données en train et test
- Transformer le dictionnaire de text et label en dataframe

Deuxième méthode

Analyse de données

- Travailler sur un sous-échantillon de l'ensemble d'apprentissage.

- La taille du sous-échantillon est K = 7000.

Deuxième méthode

Analyse de données

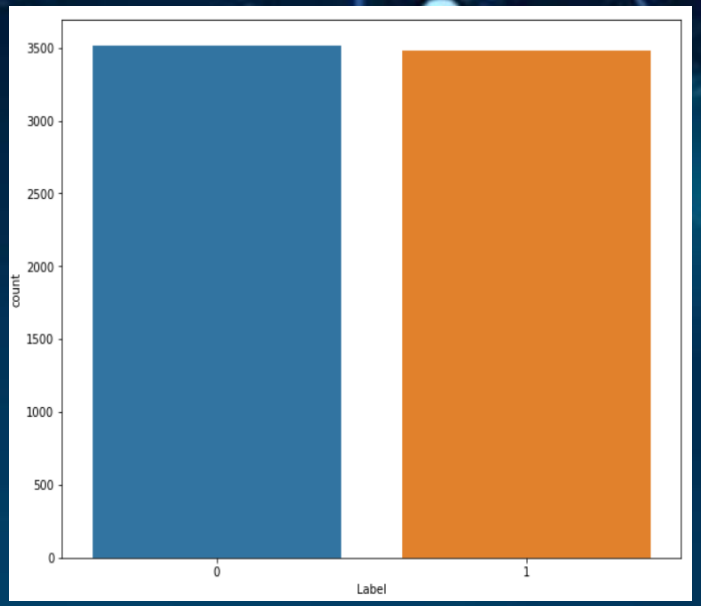
Statistique obtenues sans Lémmatisation, nettoyage et tokénisation

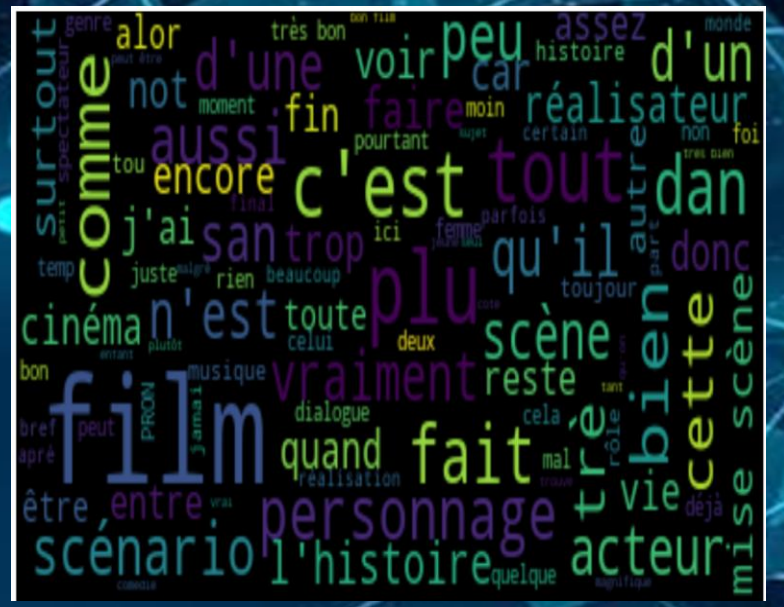
Deuxième méthode

Analyse de données

Train :

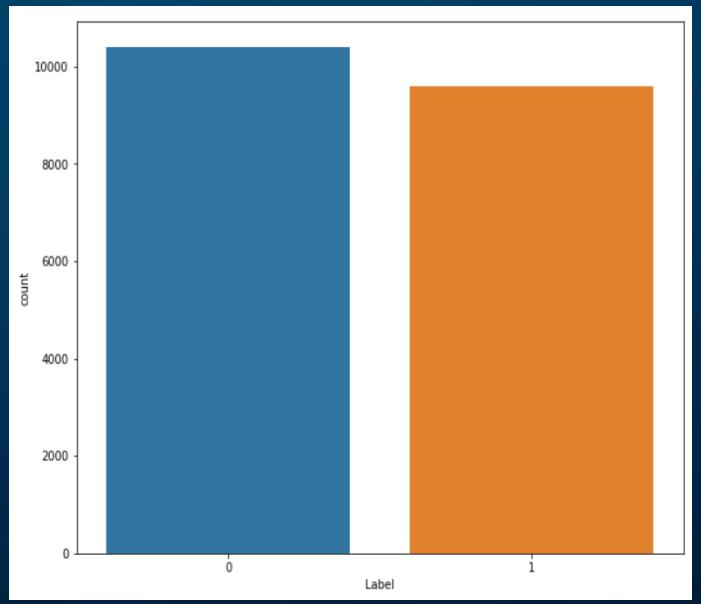
col_0	count
Label	
0	3518
1	3482





Test:

col_0	count
Label	
0	10408
1	9592



Deuxième méthode

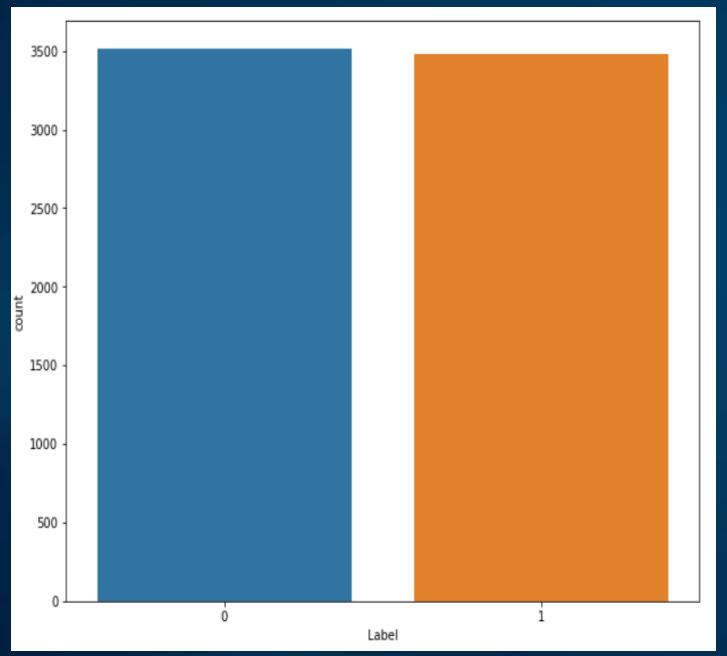
Analyse de données

Tokeniser, nettoyer et lemmatiser grâce à Spacy.

Deuxième méthode

Analyse de données

Fréquence de distribution et nuage de mots du train



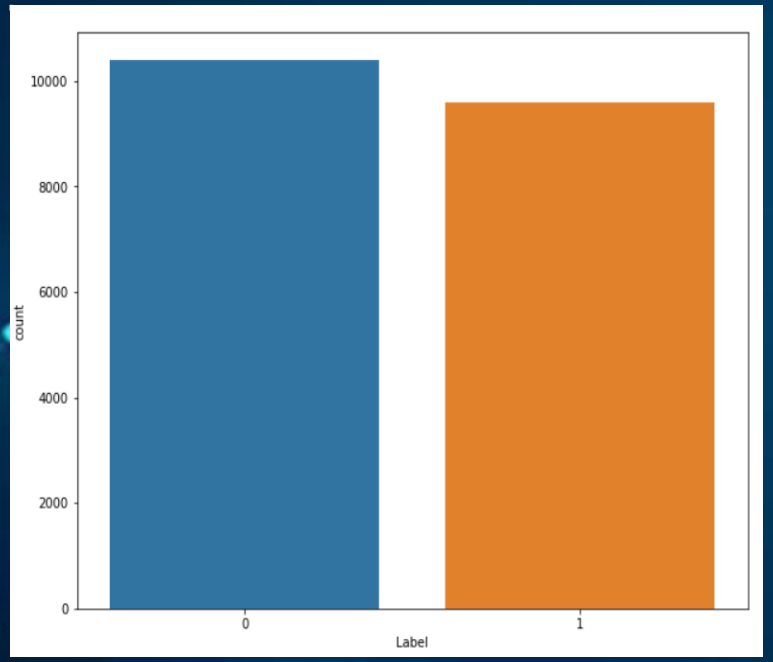
col_0	count
Label	
0	3518
1	3482

La classe la plus fréquente dans le train est 0 (négatif) 3518 critiques.

Deuxième méthode

Analyse de données

Fréquence de distribution et nuage de mots du test



col_0	count
Label	
0	10408
1	9592

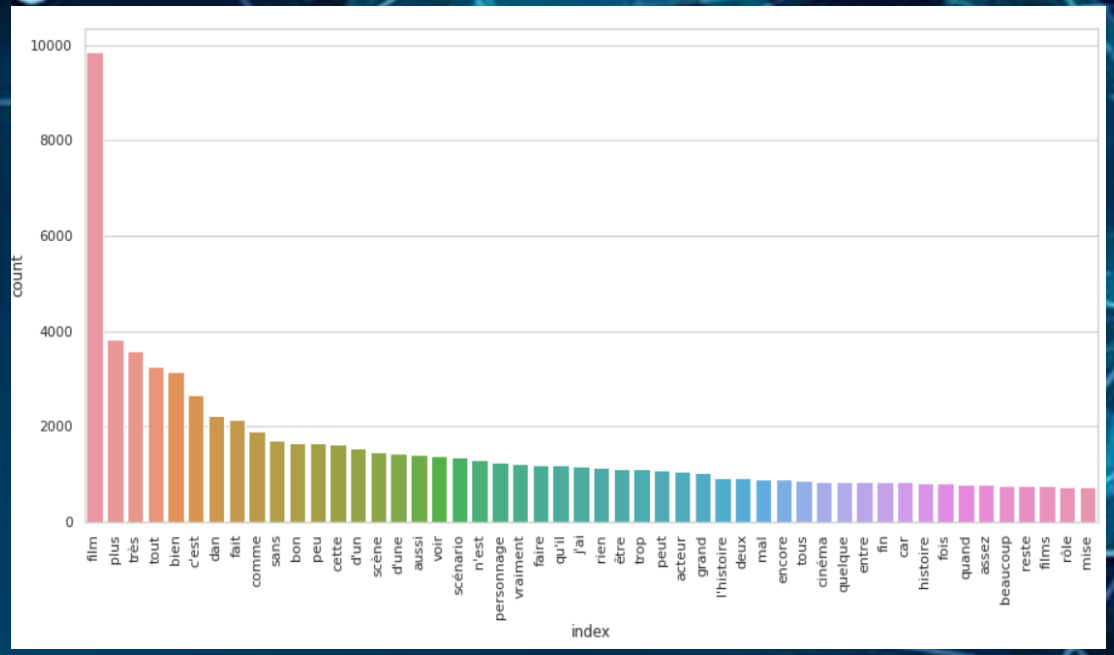
La classe la plus fréquente dans le test est 0 (négatif) 10408 critiques.

Deuxième méthode

Analyse de données

Train

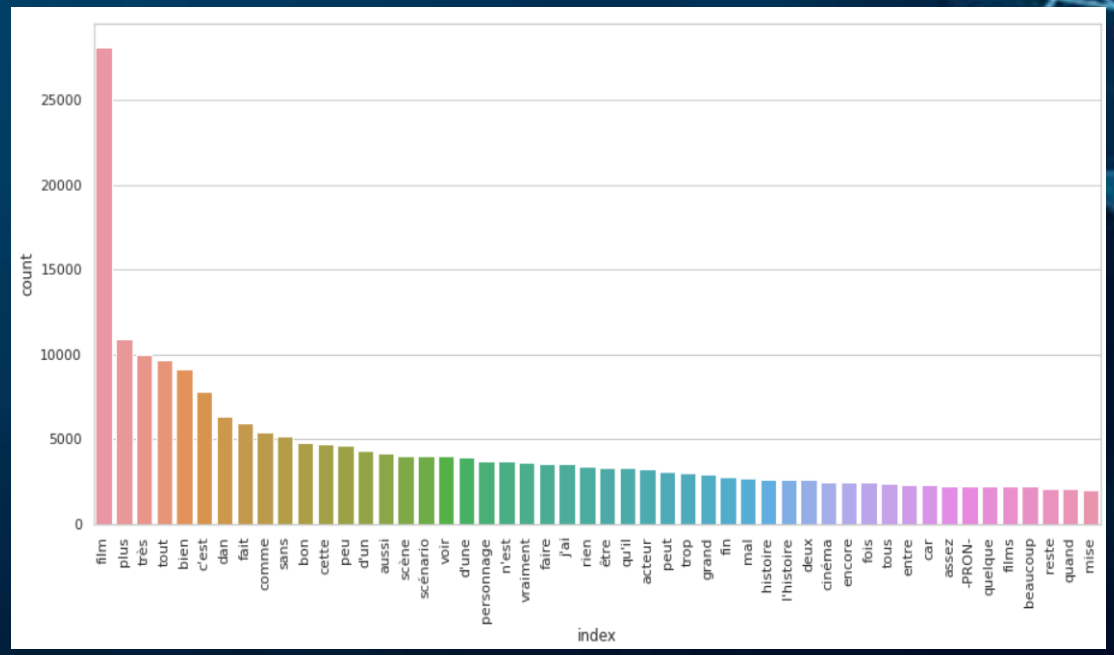
Top 50 mots des deux ensembles



Les mots les plus fréquents et communs aux deux ensembles :

- film
- plus
- très
- tout
- bien

Test



Deuxième méthode

Analyse de données

- Créer des ensembles d'apprentissage et de validation à partir de train.
- Les proportions sont bien respectées.

```
Train set:
col_0     count
Label
0      0.502476
1      0.497524
Validation set:
col_0     count
Label
0      0.502857
1      0.497143
```

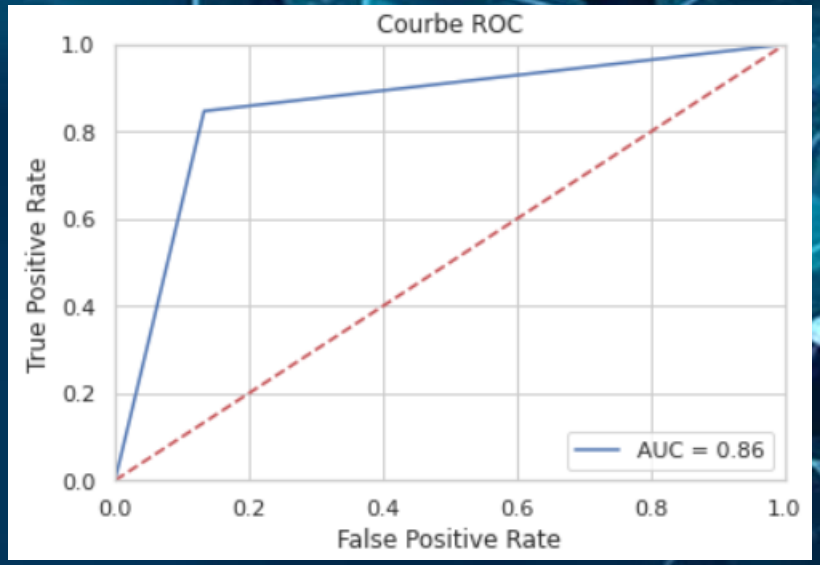
Deuxième méthode

Algorithme et Evaluation du jeu de test

Modèle 1: Logistic regression

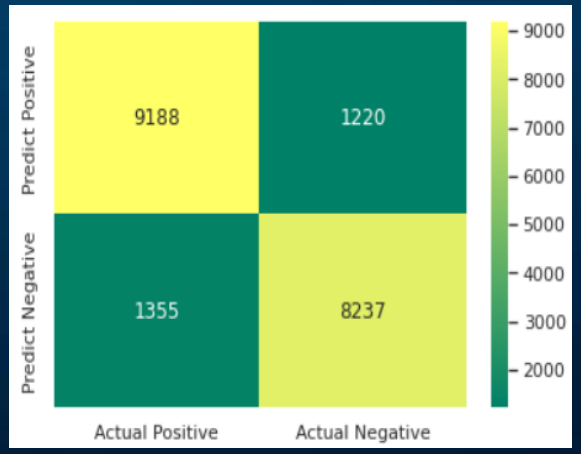
Validation

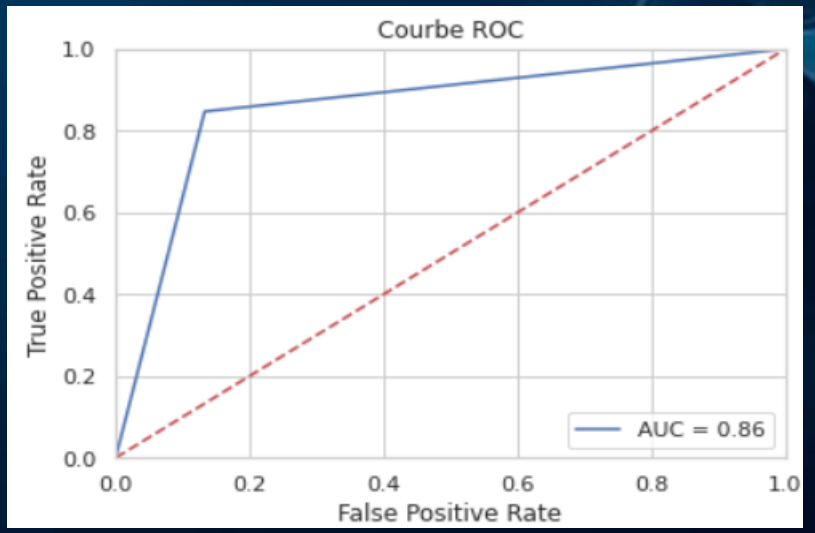
```
recall <----> 0.8471264367816091
precision <----> 0.8629976580796253
F1-Score <----> 0.8549883990719258
accuracy rate <----> 0.8571428571428571
```



Test

```
recall <----> 0.8697873227689742
precision <----> 0.8752622744439782
F1-Score <----> 0.8725162099979085
accuracy rate <----> 0.8781
```





Deuxième méthode

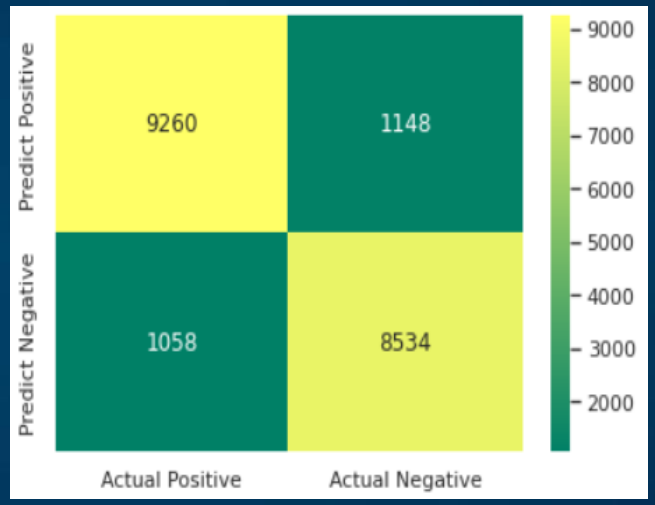
Algorithme et Evaluation du jeu de test

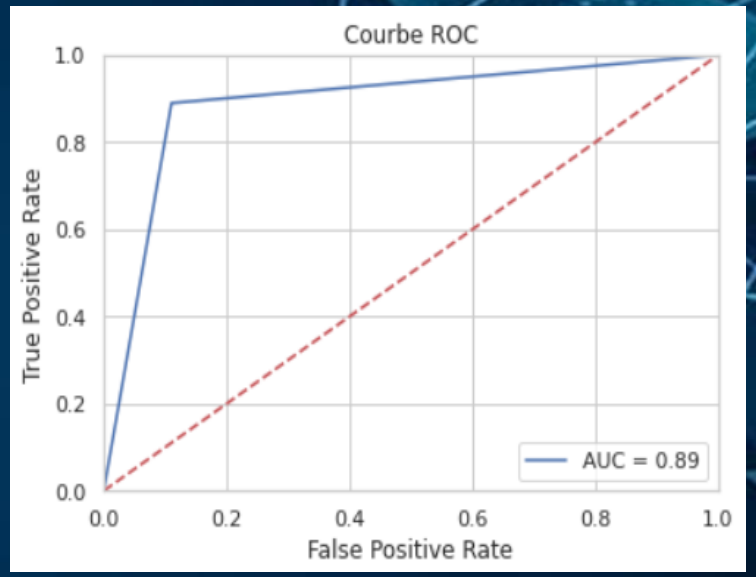
Modèle 2: Logistic Regression avec TFIDF

Test:

	precision	recall	f1-score	support
0	0.90	0.89	0.89	10408
1	0.88	0.89	0.89	9592
accuracy			0.89	20000
macro avg	0.89	0.89	0.89	20000
weighted avg	0.89	0.89	0.89	20000

Accuracy: 0.8897





On diminue le nb de features.

Deuxième méthode

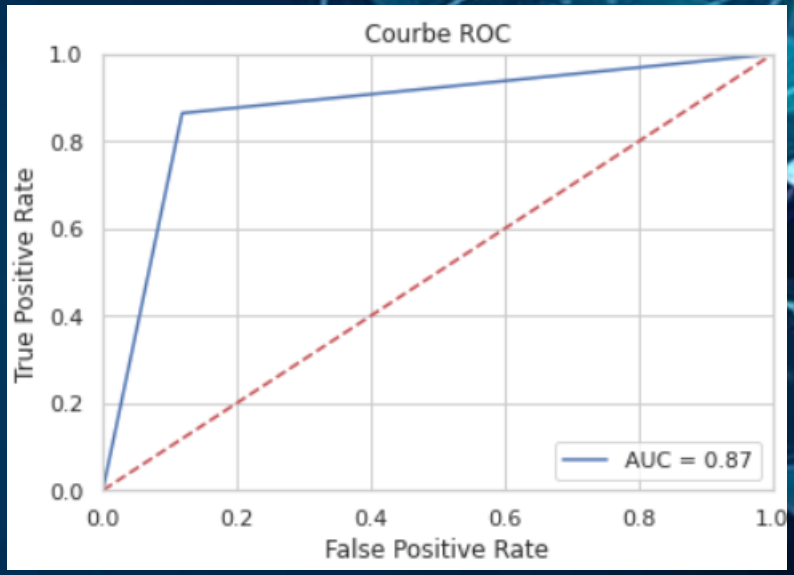
Algorithme et Evaluation du jeu de test

Modèle 2: Logistic Regression avec TFIDF

Test:

	precision	recall	f1-score	support
0	0.88	0.88	0.88	10408
1	0.87	0.86	0.87	9592
accuracy			0.87	20000
macro avg	0.87	0.87	0.87	20000
weighted avg	0.87	0.87	0.87	20000

Accuracy: 0.8734



- Le nb de features a largement diminué, mais les prédictions du modèle sont légèrement moins bonnes.
- Les coûts computationnels étant une problématique importante, on est alors gagnant.

Deuxième méthode

Algorithme et Evaluation du jeu de test

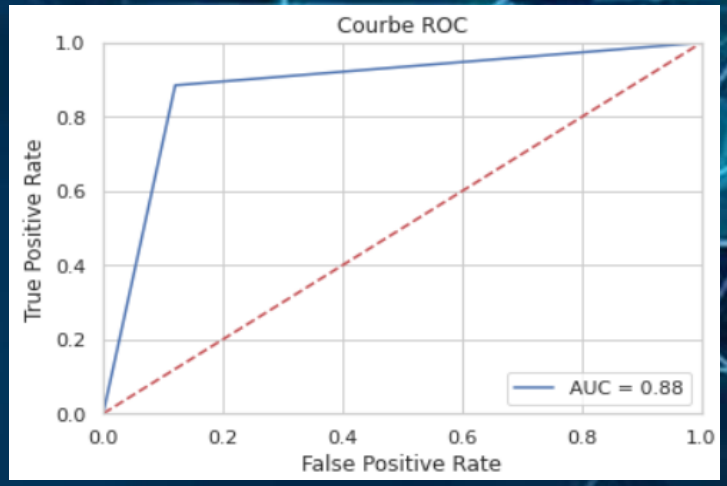
Modèle 3: Naïve Bayésien avec Vectoriseur TfIdf

Validation:

Evaluation sur validation:

	precision	recall	f1-score	support
0	0.89	0.88	0.88	880
1	0.88	0.89	0.88	870
accuracy			0.88	1750
macro avg	0.88	0.88	0.88	1750
weighted avg	0.88	0.88	0.88	1750

Accuracy: 0.8822857142857143

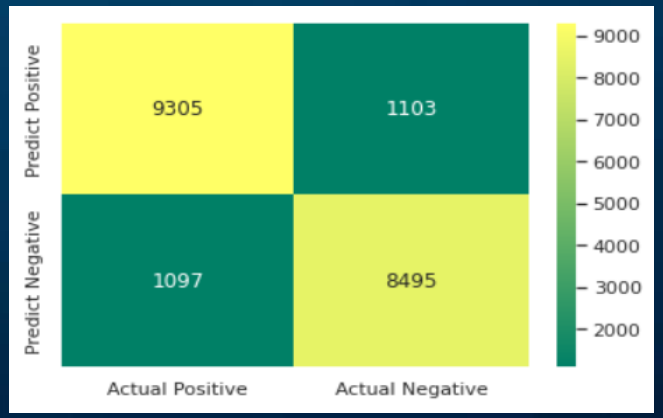


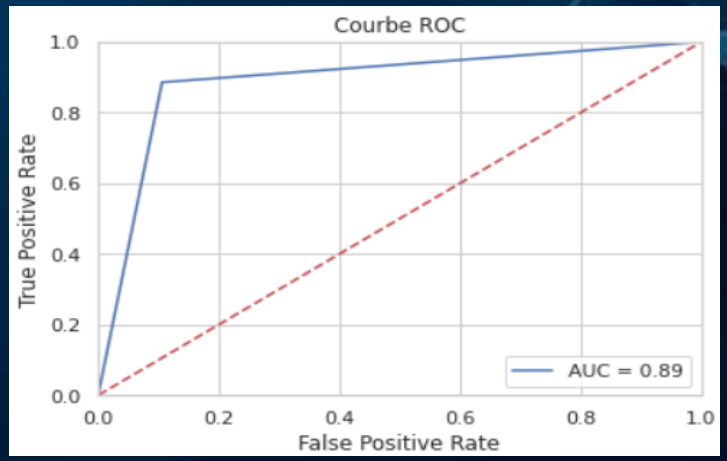
Test:

Evaluation sur test:

	precision	recall	f1-score	support
0	0.89	0.89	0.89	10408
1	0.89	0.89	0.89	9592
accuracy			0.89	20000
macro avg	0.89	0.89	0.89	20000
weighted avg	0.89	0.89	0.89	20000

Accuracy: 0.89





Les performances de ce modèle sont légèrement meilleures que les précédents, on a ici moins de faux positifs et négatifs

Deuxième méthode

Algorithme et Evaluation du jeu de test

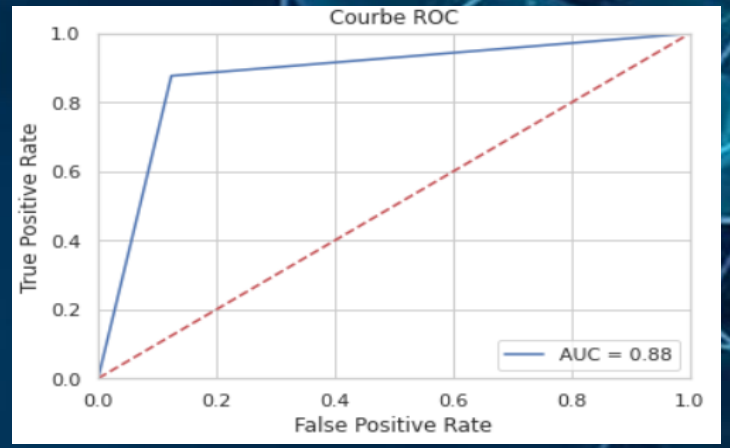
Modèle 4: Naïve Bayésien avec CountVectoriser

Validation:

Evaluation sur validation:

	precision	recall	f1-score	support
0	0.88	0.88	0.88	880
1	0.88	0.88	0.88	870
accuracy			0.88	1750
macro avg	0.88	0.88	0.88	1750
weighted avg	0.88	0.88	0.88	1750

Accuracy: 0.8765714285714286

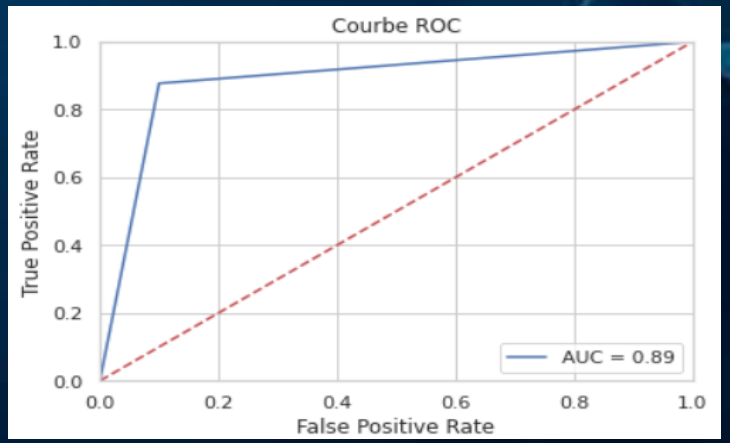


Test:

Evaluation sur test:

	precision	recall	f1-score	support
0	0.89	0.90	0.89	10408
1	0.89	0.88	0.88	9592
accuracy			0.89	20000
macro avg	0.89	0.89	0.89	20000
weighted avg	0.89	0.89	0.89	20000

Accuracy: 0.88905



La diminution des performances est très légère.

Deuxième méthode

Algorithme et Evaluation du jeu de test

Modèle 4: Naïve Bayésien avec CountVectoriser

- Testons 2 exemples de critique au hasard et regardons si elle est bien catégorisée

```
review_test = ["ce film est très intéressant, les acteurs livrent de belles performances"]

#On veut savoir comment est vectorisé cet exemple avec le transformer qu'on a initié toute à l'heure
mat_reviewTest = vect2.transform(review_test)
```

```
#Mais de quels termes s'agit-il ?
print(np.asarray(vect2.get_feature_names())[mat_reviewTest.indices])
```

```
['acteurs' 'belles' 'film' 'intéressant' 'livrent' 'performances' 'très']
```

```
predicted2 = NB_clf2.predict(mat_reviewTest)
print(predicted2)
```

```
[1]
```

```
review_test = ["le scénario n'était pas à la hauteur, tout comme le rythme trop mou de l'ensemble du film"]

#On veut savoir comment est vectorisé cet exemple avec le transformer qu'on a initié toute à l'heure
mat_reviewTest = vect2.transform(review_test)
print(mat_reviewTest)

print(np.asarray(vect2.get_feature_names())[mat_reviewTest.indices])

predicted2 = NB_clf2.predict(mat_reviewTest)
print(predicted2)
```

```
['comme' 'film' 'hauteur' "l'ensemble" 'mou' "n'était" 'rythme' 'scénario'
 'tout' 'trop']
[0]
```

Nos deux exemples ont bien été catégorisés.

Deuxième méthode

Algorithme et Evaluation du jeu de test

Modèle 5: SVM

Test:

	precision	recall	f1-score	support
0	0.89	0.89	0.89	10408
1	0.88	0.89	0.88	9592
accuracy			0.89	20000
macro avg	0.89	0.89	0.89	20000
weighted avg	0.89	0.89	0.89	20000

Accuracy: 0.8863

Pas d'augmentation conséquente

Deuxième méthode

Algorithme et Evaluation du jeu de test

Deep Neural Networks

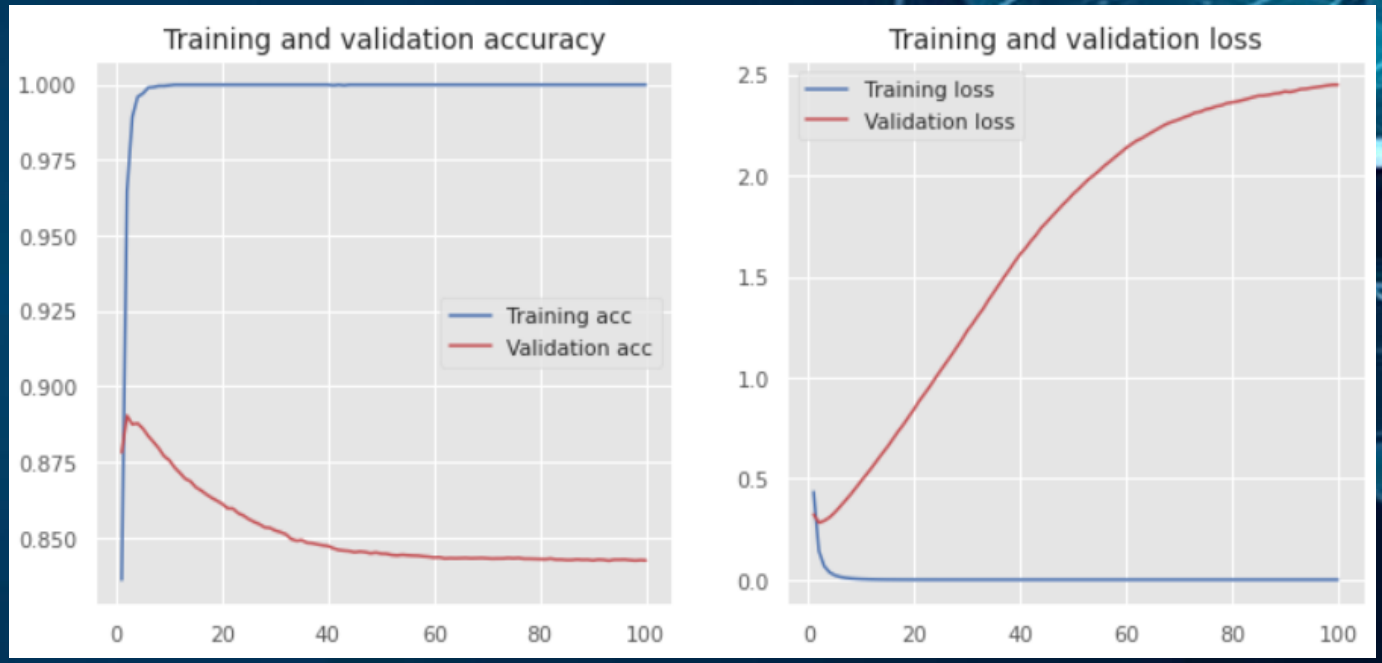
Modèle 6: Keras Model

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_26 (Dense)            (None, 10)                312640

 dense_27 (Dense)            (None, 1)                 11

=================================================================
Total params: 312,651
Trainable params: 312,651
Non-trainable params: 0
```

```
Training Accuracy: 0.9998
Testing Accuracy:  0.8424
```



- ✓ Les performances du modèle sont en dessous de ce qu'on prévoyai
- ✓ Mais pourtant c'est un modèle intéressant

Deuxième méthode

Algorithme et Evaluation du jeu de test

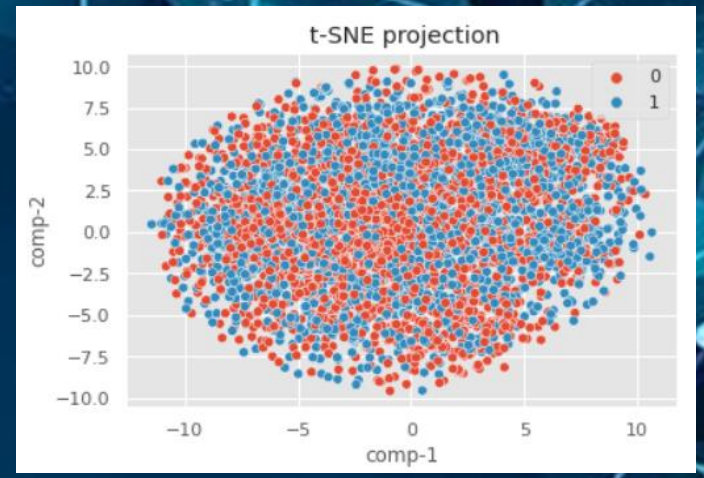
Deep Neural Networks

Modèle 7: Keras TSNE

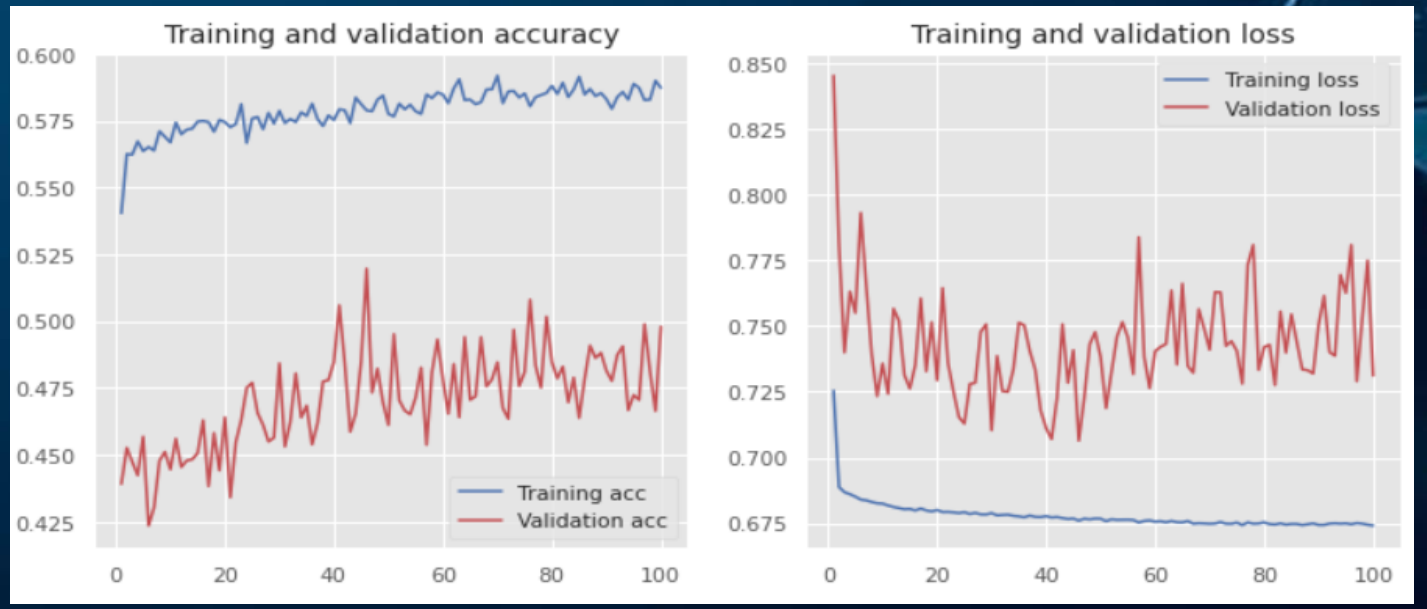
```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_24 (Dense)            (None, 10)                30

 dense_25 (Dense)            (None, 1)                 11

=================================================================
Total params: 41
Trainable params: 41
Non-trainable params: 0
```




```
Training Accuracy: 0.5899
Testing Accuracy:  0.4979
```



- Le modèle PCA n'est pas adapté à résoudre ce type de problème.
- Underfitting

Conclusion

Le choix de la méthode de transformation et modèle sera entre :

- Vectoriseur TfIdf et un Naïve Bayésien (version multinomial).
- Logistic Regression mais en faisant un TFIDF.

Le choix de la méthode de transformation et du modèle est très important pour la construction d'un bon modèle de classification.