



# Die Hard

Classification task report

---

## Team members:

Khaled Ihitt

Ousmane Assani

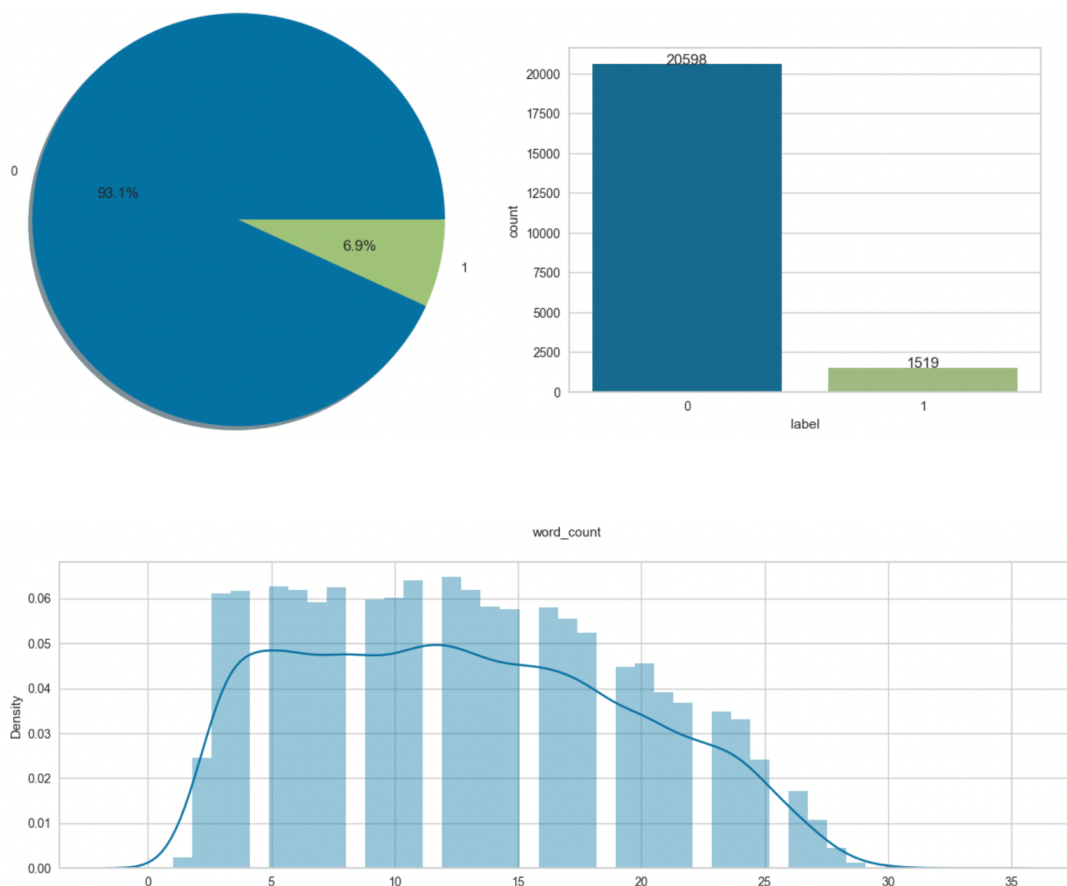
Amine Rhayem

Sirine Jlassi

## Sumerry

D'abord, nous importons les données et faisons une visualisation des données avec **pandas\_profiling** pour obtenir une vue dense des données, après nous commençons le pré-traitement des données en les étiquetant avec un simple filtre (Where) et nous supprimons la ligne dupluque.

Ci-dessous la répartition des étiquettes et la densité des mots:



de plus, nous appliquons toutes les méthodes de nettoyage de texte  
(Tokenizer, Lémétisation, Stop word)

La dernière étape avant d'appliquer les modèles de classification on a vectoryzer les donne  
et word Embedding.

## Machine learning:

List des model utiliser:

- **Naive Bayes**
- **Linear Classifier avec LogisticRegression**
- **KNeighborsClassifier**
- **DecisionTreeClassifier**
- **SVM**
- **RandomForestClassifier**
- **Extereme Gradient Boosting**

Pour chaque model on a utiliser les methode de vectorisation, Word Embeding:

- **Count Vectors**
- **Word Level TF IDF Vectors**
- **ngram level tf-idf**
- **Character Level TF IDF Vectors**

Validation Raport pour le melleur model avec les different methodes de vectorisations er Word Embeding:

**Parament les différents modèles ont des résultats très proches (91% – 93%~)**

**Gradient Boosting (Count Vectors: 93.2%~ ):**

## Count Vectors:

	precision	recall	f1-score	support
0	0.99	0.94	0.96	5438
1	0.13	0.54	0.21	92
accuracy			0.93	5530
macro avg	0.56	0.74	0.59	5530
weighted avg	0.98	0.93	0.95	5530

## WordLevel TF-IDF:

	precision	recall	f1-score	support
0	0.99	0.94	0.96	5442
1	0.11	0.47	0.17	88
accuracy			0.93	5530
macro avg	0.55	0.70	0.57	5530
weighted avg	0.98	0.93	0.95	5530


## CharLevel Vectors:

	precision	recall	f1-score	support
0	0.99	0.94	0.96	5426
1	0.12	0.45	0.19	104
accuracy			0.93	5530
macro avg	0.56	0.69	0.58	5530
weighted avg	0.97	0.93	0.95	5530

## CharLevel Vectors:

	precision	recall	f1-score	support
0	0.99	0.94	0.96	5426
1	0.12	0.45	0.19	104
accuracy			0.93	5530
macro avg	0.56	0.69	0.58	5530
weighted avg	0.97	0.93	0.95	5530

**\*\* Accuracy score**



```
Xgb, Count Vectors: 93.2007233273056 %  
Xgb, WordLevel TF-IDF: 92.94755877034359 %  
Xgb, CharLevel Vectors: 92.875226039783 %  
Xgb, CharLevel Vectors: 92.875226039783 %
```

## **\*\* Confusion matrix**

```
LR, Count Vectors:  
[[5094  335]  
 [  52   49]]
```

```
LR, WordLevel TF-IDF:  
[[5141  373]  
 [   5   11]]
```

```
LR, N-Gram Vectors:  
[[5144  382]  
 [   2    2]]
```

```
LR, CharLevel Vectors:  
[[5141  376]  
 [   5    8]]
```

## Deep learning:

List des model utiliser:

- **Recurrent Neural Network – LSTM**
- **Recurrent Neural Network – GRU**
- **Bidirectional RNN**
- **Recurrent Convolutional Neural Network**

(Avec pad\_sequences on a convertir le texte en séquence de jetons et les remplir pour garantir des vecteurs de longueur égale et applique les different algorithm)

// nombre epoque 10 pour tou les model

### Structure des models:

#### 1 - Recurrent Neural Network – LSTM:

Simple Input Layer avec les max size de phrase a chaque fois

Word embedding Layer

LSTM Layer(100 n)

Output Layers avec activation function "sigmoid"

Compile le model avec Adam pour un optimizer

Binary\_crossentropy comme loss function

#### 2- Recurrent Neural Network – GRU

La meme structure avec GRU Layer en centre

#### 3- Bidirectional RNN

La meme structure avec Bidirectional GRU Layer en centre

Et un Dropout Layer

#### 4- Recurrent Convolutional Neural Network

embedding Layer avec SpatialDropout1D

recurrent layer Bidirectional GRU

convolutional Layer Convolution1D

pooling Layer GlobalMaxPool1D

output Layers (Dense, Dropout ,Dense)

Compile avec optimizers Adam et binary\_crossentropy comme loss function

## Conclusion

La plupart des algorithmes d'apprentissage deep learning mettent un surajustement pré 10 epoc, alors nous avons conclu que le problème est que les données ne son pas équilibrées en termes de classe (le coût de l'étiquette 0 prend la majorité) .

Donc dans notre cas le meilleur résultat est d'obtenir avec un modèle de machine Learning.