République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Tunis El Manar
Ecole Nationale d'Ingénieurs de Tunis

Rapport de module « Deep Learning »

Sentiment Analysis using NLP

Encadré par :

Mme Sabrine KRICHEN

M. Fares EL KAHLA

Effectué par :

Chaima HAJTAHER & Ranim TAKTAK

Classe :

3A DASSEC

Année universitaire 2022/2023

Remerciements

Tout d'abord, Je tiens à remercier tous ceux qui m'ont aidé dans ce travail en me fournissant les informations nécessaires à la réussite de ce projet.

Je tiens également à exprimer ma gratitude à nos deux superviseurs, **M. Fares El Kahla** et **Mme Sabrine Krichen,** qui nous avons fourni des conseils tant au niveau théorique que pratique et m'a permis de bénéficier de son expérience.

Table des figures

Introduction générale

L’analyse des sentiments est le processus qui consiste à analyser un texte numérique pour déterminer si ton émotionnel du message est positif, négatif ou neutre . Aujourd’hui, les entreprises disposent d’importants volumes de données textuelles telles que des e-mails, des transcriptions, des conversations, des commentaires sur les réseaux sociaux et des avis.

Dans ce rapport, nous allons présenter les différentes phases de l’implémentation de notre solution « Sentiment analysis ».

I. Introduction :

Dans le cadre d'évaluation de module « Deep learning », nous allons développer un modèle pour l'analyse de sentiments en utilisant le NLP « Le traitement de langages naturels ».

II. L'environnement de travail :

1. Représentation d'outils utilisées :

1.1 La partie hardware :

Pour notre projet, nous avons utilisé un ordinateur portable MSI avec les caractéristiques suivantes :

Processeur (CPU)	Intel core-I5
Mémoire (RAM)	8.00 Go
Système d'exploitation	Windows 10, 64 bits
Ecran	15.6

Tableau 1 : La partie hardware de l'environnement de travail

1.2 La partie software :

1.2.1 Le langage de programmation utilisé :

Après des recherches approfondies, il a été conclu que le langage de programmation le plus adapté à notre projet était Python, qui appartient au domaine de l'intelligence artificielle.

En fait, Python est un langage de script de haut niveau, structuré et open source. Il est multi paradigme et multitâche et polyvalent. Ce langage de programmation présente de nombreuses caractéristiques intéressantes, telles que

- Gratuit : Python est gratuit, mais il peut être utilisé pour des projets commerciaux sans restriction.
- Simplicité : Python est relativement facile à apprendre.
- Python convient pour des scripts d'une douzaine de lignes ou pour des projets complexes de plusieurs dizaines de milliers de lignes.
- Python gère ses ressources (mémoire, descripteurs de fichiers .....) sans l'intervention d'un programmeur.

- Python est largement utilisé en intelligence artificielle et, plus généralement, en analyse de données. Toutes ces caractéristiques font de Python l'outil idéal pour la mise en œuvre de nos applications.

1.2.2 Les bibliothèques utilisées :

Pour arriver à réaliser notre projet, nous avons utilisées plusieurs bibliothèques. Parmi d'eux, on peut citer :

- **Numpy :**

Numpy est la bibliothèque de base pour le calcul scientifique en Python. Il fournit un objet tableau multidimensionnel très puissant et des outils pour travailler avec ces tableaux. La bibliothèque est fortement optimisée pour les opérations numériques et sa syntaxe est proche de celle de MATLAB.

Le principe de base de Numpy est de fournir des tableaux multidimensionnels. En outre, NumPy peut être considéré comme la base du calcul matriciel en Python, qui est conçu à des fins mathématiques et scientifiques.

- **Pandas :**

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations et des opérations de manipulation de tableaux numériques et des séries temporelles.

- **Tensorflow :**

Tensorflow est une bibliothèque d'apprentissage automatique, une boîte à outils permettant de résoudre facilement des problèmes mathématiques extrêmement complexes.

La bibliothèque peut être utilisée pour former et exécuter des réseaux neuronaux pour la classification de chiffres manuscrits, la reconnaissance d'images, la suffixation de mots, les réseaux neuronaux récursifs, les modèles de séquence à séquence pour la traduction automatique et le traitement du langage naturel.

- **Seaborn :**

Seaborn est une bibliothèque permettant de créer des graphiques statistiques en Python. Elle est basée sur Matplotlib, et s'intègre avec les structures Pandas.

- **Nltk :**

Natural Language ToolKit est une bibliothèque logicielle en Python permettant un traitement automatique des langues, développée par Bird et Loper du département d'informatique de l'université de Pennsylvanie.

- **Matplotlib :**

Matplotlib est une bibliothèque du langage de programmation Python permettant de tracer et de visualiser des données sous forme de graphiques. Il peut être combiné avec les bibliothèques de calcul scientifique Python NumPy et SciPy. Il fournit également une API orientée objet pour l'intégration de graphes dans des applications utilisant des outils d'interface graphique courants tels que Tkinter, wxPython, Qt ou GTK.

- **Wordcloud :**

C'est une application en ligne pour générer des « nuage de mots » à partir d'un texte, documents ou d'une URL. Les nuages donnent plus d'importance aux mots qui apparaissent le plus fréquemment dans le texte.

1.2.3 La base de donnée utilisée :

La base de données « **the sentiment140 dataset »** a été utilisée comme notre Dataset. Cette dernière comprend un total de **1,600,000 tweets** extraite depuis the Twitter API. Elle est étiquetée avec l'une des 2 catégories d'émotions suivantes :

0: 800 000 tweets → Negative Tweets

4: 800 000 tweets → Positive tweets

Elle a 6 caractéristiques ou "Features":

- **Sentiment:** the polarity of the tweet (0 = negative, 4 = positive).
- **Id:** The id of the tweet.
- **Date:** the date of the tweet.
- **Flag:** The query (lyx). If there is no query, then this value is NO_QUERY.

- **User:** the user that tweeted.
- **Text:** the text of the tweet**.**

III. Implémentation de modèle d'analyse de sentiments :

1. Importation des bibliothèques :

Dans cette partie, on va importer les bibliothèques nécessaires qu'on va utiliser tout au long de projet. La figure 1 représente les bibliothèques utilisées.

```
import os
import numpy as np
import pandas as pd
import tensorflow as tf
import seaborn as sns
import re
import matplotlib.pyplot as plt
import nltk
```

Figure 1 : Importation des bibliothèques

2. Manipulation du dataset :

Au premier lieu, on va surcharger la dataset pour avoir une vision globale sur la description de notre base de données (les caractéristiques, les nombres d'observations,). En fait, notre base de donnée comporte 1 600 000 observations et 6 attributs ou caractéristiques.

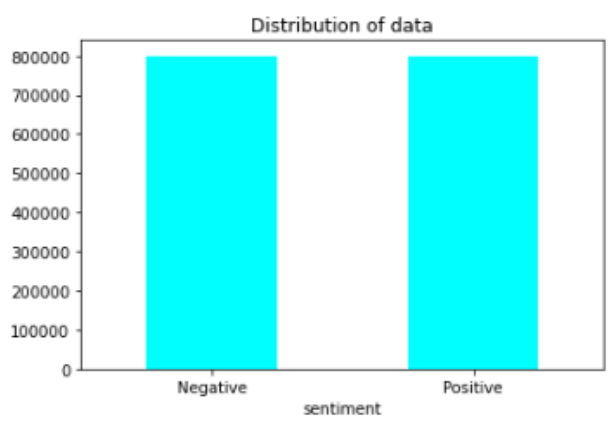
	sentiment	id	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
5	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew
6	0	1467811592	Mon Apr 06 22:20:03 PDT 2009	NO_QUERY	mybirch	Need a hug
7	0	1467811594	Mon Apr 06 22:20:03 PDT 2009	NO_QUERY	coZZ	@LOLTrish hey long time no see! Yes.. Rains a...
8	0	1467811795	Mon Apr 06 22:20:05 PDT 2009	NO_QUERY	2Hood4Hollywood	@Tatiana_K nope they didn't have it
9	0	1467812025	Mon Apr 06 22:20:09 PDT 2009	NO_QUERY	mimismo	@twittera que me muera ?

```
print(f"The dataset contains {dataset.shape[0]} observations, from which we have {dataset.shape[1]} attributes")
```

```
The dataset contains 1600000 observations, from which we have 6 attributes
```

Figure 2 : Importation du dataset

Puis, on a appliqué certaines bibliothèques Python pour voir la distribution de la base de donnée comme, par exemple, pour savoir si la base de données contient des valeurs nulles ou non, la base de donnée est équilibrée ou non. La figure 3 représente la distribution de data.




```
Column    Missing values
----------------------------

sentiment    0
id           0
date         0
flag         0
user         0
text         0
dtype: int64
```

Figure 3 : La distribution de data

Pour l'analyse de sentiments, on a besoin de deux attributs seulement qui sont {Text, sentiment}. C'est pour cela qu'on a procédé de supprimer les autres colonnes. Aussi, on a remplacé le label 4 qui correspond au positive tweets par le label 1 à fin d'avoir une « Binary Classification ».

La figure 4 représente les opérations faites.

```
dataset['sentiment'] = dataset['sentiment'].replace(4,1)
```

```
#checking feature's type
dataset.dtypes
```

```
sentiment     int64
text         object
dtype: object
```

Figure 4 : La dataset après certains changements

3. Text processing :

Pour la partie « Text processing », on va appliquer les techniques suivantes :

- Lire " **contractions.csv** " and la stocker comme un dictionnaire. La figure 5 représente les dix premières lignes de « Contractions.csv ». Ce dernier contient 148 lignes au total.

Contraction	Meaning
'aight	alright
ain't	is not
amn't	am not
aren't	are not
can't	cannot
'cause	because
could've	could have
couldn't	could not
couldn't've	could not have
daren't	dare not

Figure 5 : les dix premières lignes de « Contractions.csv »

- On a remplacé tous les liens URL avec '<url>'.
- On a remplacé tous les @USERNAME to '<user>'.
- On a remplacé les 3 lettres consécutifs ou plus par deux lettres.
- On a supprimé les symboles et les caractéres non-alphanumérique.
- On a remplacé tous les émojis.
- On a ajouté un espace ‘/’ pour la séparation des mots (Après le remplacement des URLS).

La figure 6 représente le résultat des changements déjà étudiés ci-dessus.

	sentiment	text	processed_text
0	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...	<user> <url> aww thatis a bummer you shou...
1	0	is upset that he can't update his Facebook by ...	is upset that he cannot update his facebook by...
2	0	@Kenichan I dived many times for the ball. Man...	<user> i dived many times for the ball manage...
3	0	my whole body feels itchy and like its on fire	my whole body feels itchy and like its on fire
4	0	@nationwideclass no, it's not behaving at all....	<user> no it is not behaving at all i am mad...

Figure 6 : Processed_text

4. Analyse de dataset

Dans cette partie, on a appliqué « Wordcloud » sur les positive tweets et les négative tweets pour savoir les mots qui apparaissent le plus fréquemment dans le texte.

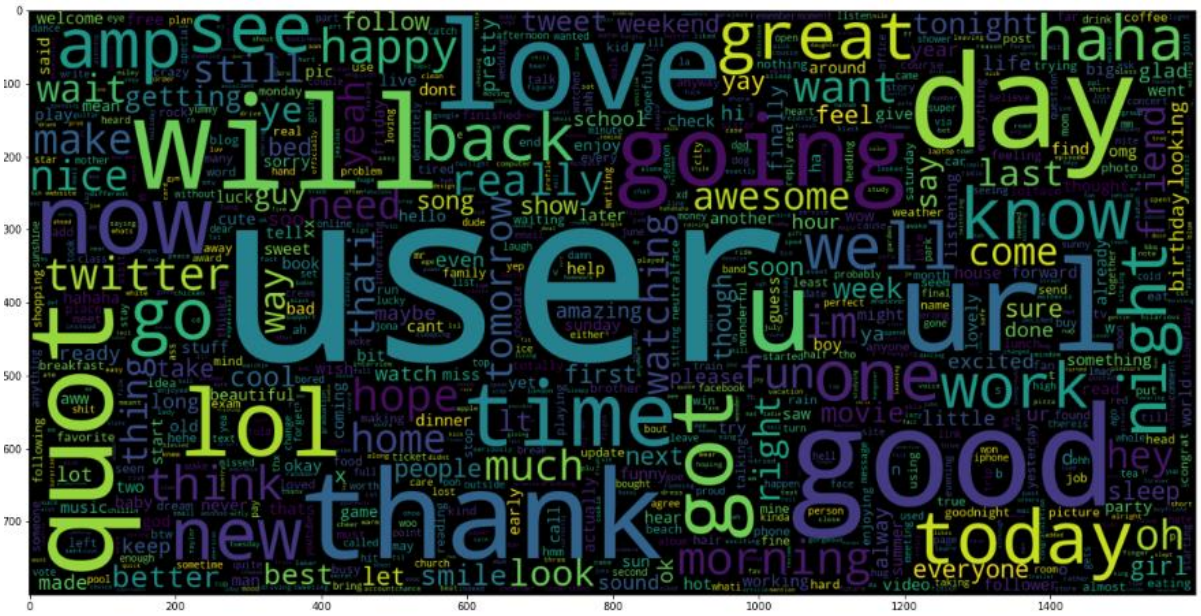


Figure 7: Wordcloud generation for positive tweets

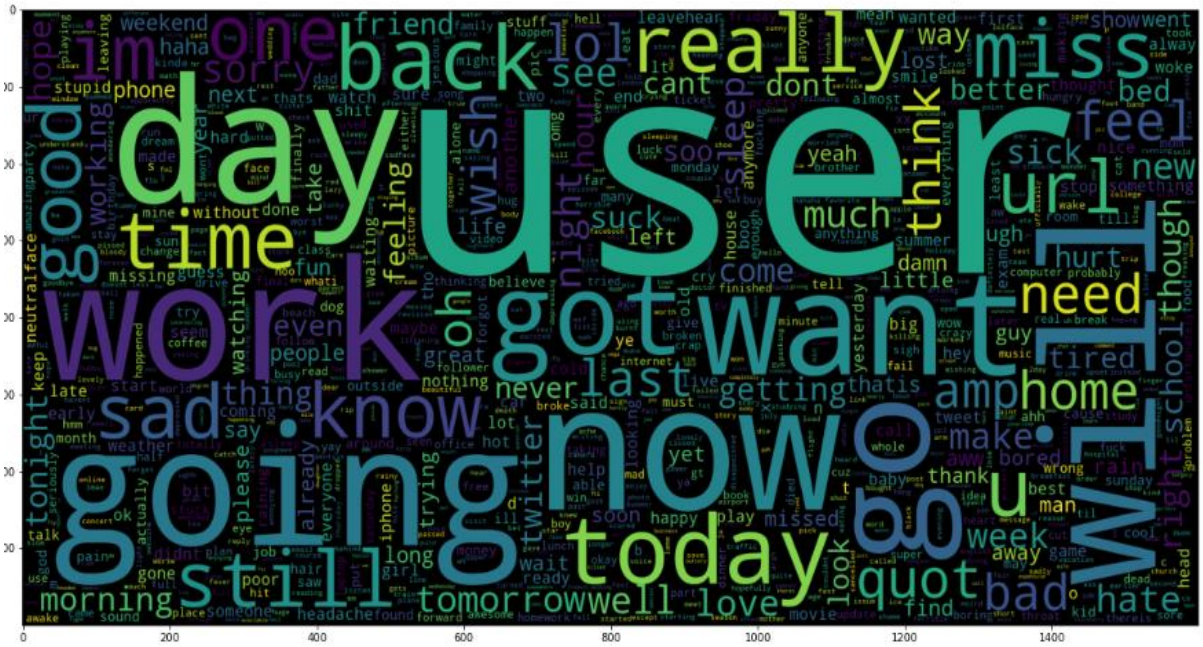


Figure 8: Wordcloud generation for negative tweets

5. Division de Dataset :

On a choisi de consacrer 80 % de données pour la partie « Training » et 20% de données pour la partie test comme présenter dans la figure 9.

```
from sklearn.model_selection import train_test_split
X_data, y_data = np.array(dataset['processed_text']), np.array(dataset['sentiment'])
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data,test_size = 0.2, random_state = 0)
```

Figure 9: Division de dataset

6. Word Embeddings :

La dernière étape dans la préparation de données est la conversion des mots en des nombres parce que l'ordinateur ne peut pas comprendre le Text. C'est pourquoi, on a cherché une solution pour mettre la définition de chaque en un vecteur.

On a trouvé la fonction Word2Vec () qui permet la création et l'entrainement de « word Embeddings » en utilisant notre dataset.

```
Word2vec_train_data[0]
```

```
['<user>', 'thx', 'i', 'like', 'ur', 'pic', 'as', 'well']
```

```
# Defining the model and training it.
word2vec_model = Word2Vec(Word2vec_train_data, vector_size=Embedding_dimensions, workers=8, min_count=5)
```

```
print("Vocabulary Length:", len(word2vec_model.wv.key_to_index))
```

```
Vocabulary Length: 47149
```

Figure 10 : Word Embeddings

7. Tokenization :

Cette étape permet de séparer les textes en des Units appelé "tokens". Les tokens peuvent être des caractéres, des mots ou n-gram caractéres.

```
vocab_length = 60000

tokenizer = Tokenizer(filters="", lower=False, oov_token="<oov>")
tokenizer.fit_on_texts(X_data)
tokenizer.num_words = vocab_length
print("Tokenizer vocab length:", vocab_length)
```

```
Tokenizer vocab length: 60000
```

Figure 11 : Tokenization

Embeddings Matrix est une matrice qui contient tous les mots et leur correspondance embeddings. On va l'utiliser ultérieurement dans la couche « Embeddings Layer ».

```
embedding_matrix = np.zeros((vocab_length, Embedding_dimensions))

for word, token in tokenizer.word_index.items():
    if word2vec_model.wv.__contains__(word):
        embedding_matrix[token] = word2vec_model.wv.__getitem__(word)

print("Embedding Matrix Shape:", embedding_matrix.shape)
```

```
Embedding Matrix Shape: (60000, 100)
```

```
X_train.shape
```

```
(1280000, 60)
```

Figure 12 : Embeddings Matrix

8. Création de modèle :

Au premier lieu, on a créé un modèle simple avec seulement trois couches embedding_6, Flatten et dense_7}.

```
training_model = Model()
training_model.summary()
```

```
Model: "Sentiment_Model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_6 (Embedding)     (None, 60, 100)           6000000

 flatten (Flatten)           (None, 6000)              0

 dense_7 (Dense)             (None, 1)                 6001

=================================================================
Total params: 6,006,001
Trainable params: 6,001
Non-trainable params: 6,000,000
_________________________________________________________________
```

Figure 13 : Model 1 Summary

Puis, on a entrainé le modèle en utilisant les paramètres suivants :

- Loss='binary_crossentropy'
- Optimizer='adam'
- Metrics=['accuracy']
- Epochs=10
- Validation_split=0.1
- Batch_size=1024
- Verbose=1
- Learning rate = 0.0010

La figure 14 représente l'entrainement du modèle 1 avec les paramètres précédents :

```
history = training_model.fit(
    X_train, y_train,
    batch_size=1024,
    epochs=10,
    validation_split=0.1,
    callbacks=callbacks,
    verbose=1,
)
```

```
Epoch 1/10
1125/1125 [==============================] - 15s 13ms/step - loss: 0.5128 - accuracy: 0.7543 - val_loss: 0.5066 - val_accuracy:
0.7627 - lr: 0.0010
Epoch 2/10
1125/1125 [==============================] - 15s 13ms/step - loss: 0.5027 - accuracy: 0.7618 - val_loss: 0.5028 - val_accuracy:
0.7624 - lr: 0.0010
Epoch 3/10
1125/1125 [==============================] - 15s 13ms/step - loss: 0.5027 - accuracy: 0.7622 - val_loss: 0.5031 - val_accuracy:
```

Figure 14 : Entrainement du « Model 1 »

Pour améliorer le modèle, on choisit d'ajouter une autre layer « Bidirectionnel(LSTM)) »
Comme présenté dans la figure 15.

```
training_model = Model()
training_model.summary()
```

```
Model: "Sentiment_Model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_15 (Embedding)    (None, 60, 100)           6000000

 bidirectional_8 (Bidirectio  (None, 60, 100)          60400
 nal)

 flatten_6 (Flatten)         (None, 6000)              0

 dense_13 (Dense)            (None, 1)                 6001

=================================================================
Total params: 6,066,401
Trainable params: 66,401
Non-trainable params: 6,000,000
_________________________________________________________________
```

Figure 15 : Model 2 Summary

La figure 16 représente l'entrainement de modèle 2.

```
history = training_model.fit(
    X_train, y_train,
    batch_size=1024,
    epochs=10,
    validation_split=0.1,
    callbacks=callbacks,
    verbose=1,
)
```

```
Epoch 1/10
1125/1125 [==============================] - 477s 423ms/step - loss: 0.4247 - accuracy: 0.8025 - val_loss: 0.4046 - val_accurac
y: 0.8152 - lr: 0.0010
Epoch 2/10
1125/1125 [==============================] - 449s 399ms/step - loss: 0.3902 - accuracy: 0.8227 - val_loss: 0.3940 - val_accurac
y: 0.8207 - lr: 0.0010
Epoch 3/10
```

Figure 16 : Entrainement de modèle 2

Pour améliorer le modèle 2, on choisit d'ajouter trois layer « Bidirectionnel(LSTM)) »

Comme présenté dans la figure 17.

```
def Model():
    embedding_layer = Embedding(input_dim = vocab_length,
                                output_dim = Embedding_dimensions,
                                weights=[embedding_matrix],
                                input_length=input_length,
                                trainable=False)

    model = Sequential([
        embedding_layer,
        Bidirectional(LSTM(50,dropout=0.2,return_sequences=True)),
        Bidirectional(LSTM(50,dropout=0.2,return_sequences=True)),
        Bidirectional(LSTM(50,dropout=0.2,return_sequences=True)),
        Flatten(),
        Dense(1, activation='sigmoid'),
    ],
    name="Sentiment_Model")
    return model
```

Figure 17 : Model 3 Summary

La figure 18 représente l'entrainement de modèle 3 en augmentant le nombre d'Epochs de 10 à 15.

```
history = training_model.fit(
    X_train, y_train,
    batch_size=1024,
    epochs=15,
    validation_split=0.1,
    callbacks=callbacks,
    verbose=1,
)
```

```
Epoch 1/15
 224/1125 [====>.....................] - ETA: 41:59 - loss: 0.4850 - accuracy: 0.7652
```

Figure 18 : Entrainement du Model 3

9. Evaluation de modèle :

L'evaluation de modèle est une étape importante dans l'apprentissage automatique. Il permet de mesurer la performance de modèle et comprendre comment il permet la généralisation en passant une autre donnée qui n'a pas vu précédemment.

9.1 Accuracy :

La métrique "Accuracy" est l'une des méthodes la plus utilisé pour l'évaluation des modèles de machine Learning et Deep Learning. La figure 19 représente l'accuracy de modèle.

```
[ ] #Testing the Trained model on test data
    accr1 = model.evaluate(X_test,Y_test) #we are starting to test the model here

    375/375 [==============================] - 3s 8ms/step - loss: 0.5556 - accuracy: 0.7356

[ ] #Accuracy
    print('Test set\n  Accuracy: {:0.2f}'.format(accr1[1])) #the accuracy of the model on test data is given below

    Test set
      Accuracy: 0.74
```

Figure 19 : Model Accuracy

9.2 Confusion Matrix :

La matrice de confusion est une table utiliser pour définir la performance d'un algorithme de classification. Il nous permet de voir les pourcentages des valeurs TP (True positive), FP (False positive), FN (False negative) et TN (True negative).

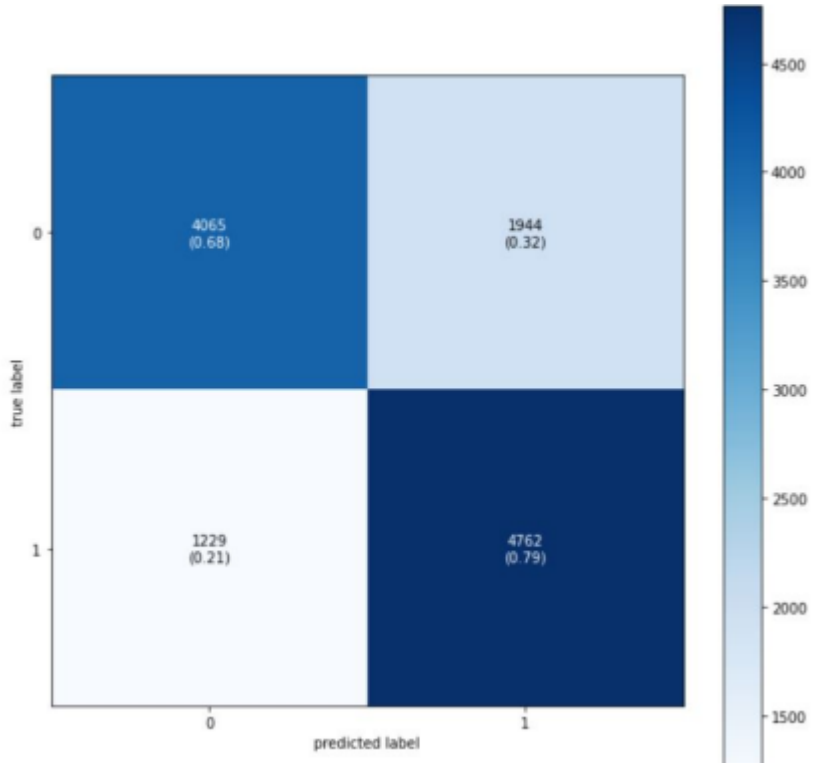


Figure 20 : Model Confusion Matrix

9.3 ROC Curve :

ROC Curve and AUC sont utilisés pour le cas d'une "Binary Classification". ROC Curve représente le pourcentage des valeurs TP contre les valeurs FP. De plus, AUC mesure les zones au-dessous de ROC Curve. Auc est une valeur entre 0 et 1. Où, la valeur 1 représente l'idéale classifier et la valeur 0 représente un aléatoire classifier.

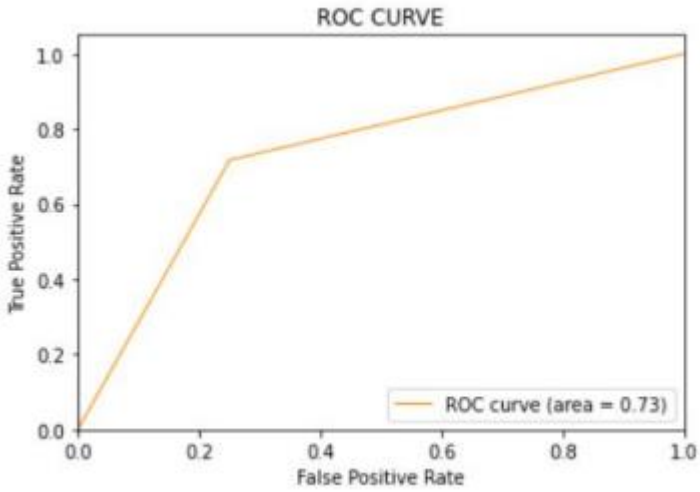


Figure 21 : Roc Curve

10.Pre-trained Model :

BERT (Bidirectional Encoder Representations from Transformers) est fait référence à un algorithme basé sur le traitement du langage naturel (NLP) et les réseaux neuronaux.

Il s'agit d'une intelligence artificielle que Google avait déjà publiée en open source à l'automne 2018. Google BERT comprend mieux le sens et la relation des mots. Avec Google Update, le moteur de recherche ne traite plus seulement des mots-clés individuels, mais plus également comprendre le contexte et la relation entre les mots.

BERT is disponible dans différends modèles pre-entrainé

10.1 BERT-Base :

BERT-Base est une version pré-entrainé de modèle BERT. IL a 12 couches ou layers (transformer blocks), 12 attention heads, and 110 millions de paramètres. BERT-Base est entrainé à travers une large corpus des données textuelles , y compris "English Wikipedia " et "the BooksCorpus dataset ''.

Dans notre cas, on va utiliser « **Bert-Base uncased** ». En effet, le modèle a été entrainé sur des données textuelles qui ont été mises en minuscule et segmentées. Sans préserver la case des caractéres. Cela signifie que le modèle a appris à traiter les mots avec une capitalisation différente comme le même mot "Apple" and "apple".

11.Enregistrement de modèle :

La derniére étape de project est l'enregistrement de modèle et les poids.

```
: training_model.save('Sentiment Analysis')
  training_model.save_weights('model weights/weights')
```

Figure 22 : Save Model

IV. Conclusion :

On a arrivé à la fin de ce projet d'implémenter un modèle d'analyse de sentiments en utilisant NLP et ses différends techniques. Comme perspective, il reste la partie de déploiement de modèle comme une application Web en utilisant Streamlit.

Conclusion générale

Dans ce rapport, nous avons implémenté un modèle de deep learning pour améliorer la performance de notre système d'analyse de sentiments. Cette tâche qui consiste à déterminer le sentiment à travers un text.

L'analyse de sentiment est un domaine de recherche en forte croissance et ne cesse qu'à innover. Ce projet est une opportunité pour nous parce que, d'une part, ce projet nous a permis de développer nos compétences dans le domaine non seulement dans le domaine de NLP « le traitement des langages naturels ». Mais, aussi, dans le domaine de Deep Learning. D'autre part, nous allons continuer à améliorer le modèle en utilisant toute la base de donnée, d'autres méthodes d'optimisation et en fin déployer le modèle