**Prédiction de « bad buzz » : 3 méthodologies différentes**

* Comment fonctionne la détéction de sentiments ?

L’analyse de sentiment est un outil d’apprentissage automatique qui analyse du texte et qui prédit si le sentiment associé est positif ou négatif.

En formant des outils d'apprentissage automatique avec des exemples d'émotions dans le texte, les machines apprennent automatiquement à détecter les sentiments sans intervention humaine.

Pour faire simple, l'apprentissage automatique permet aux ordinateurs d'apprendre de nouvelles tâches sans être expressément programmés pour les exécuter. Les modèles d'analyse des sentiments peuvent être entraînés à lire au-delà de simples définitions, à comprendre des choses comme le contexte, le sarcasme et les mots mal appliqués. Par exemple imaginons la phrase suivante :

‘‘Interface super facile d’utilisation : un diplôme d'ingénieur serait utile.’’

Hors contexte, les mots « super facile » et « utile » pourraient être lus comme positifs, mais il s'agit clairement d'un commentaire négatif (ironique).

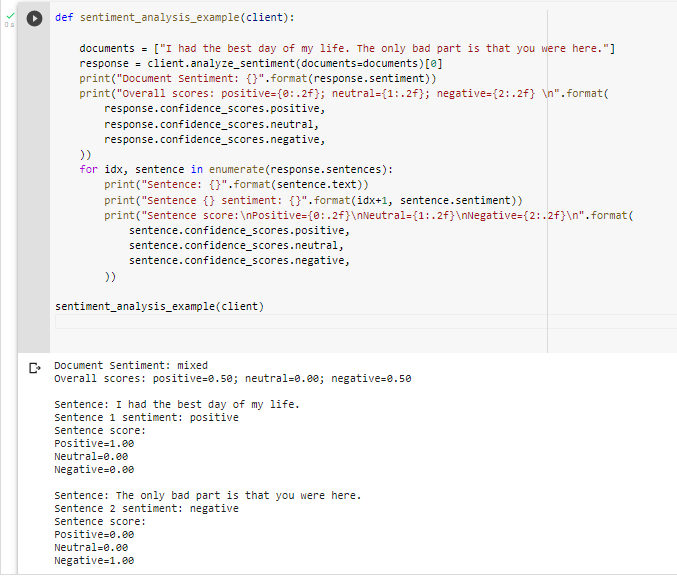
Grâce à l'analyse des sentiments, les ordinateurs peuvent traiter automatiquement les données textuelles et les comprendre comme un humain le ferait, économisant ainsi des centaines d'heures de travail.

Nous allons voir dans cet article trois méthodes de détection de sentiments sur des tweets et analyser ensemble les points positifs et négatifs de chacun.

**Méthode 1 : Api sur étagère**

Il s’agit d’un modèle déjà entrainé par Microsoft. Il y a juste a envoyer du texte via une requête API via un notebook ou du code python classique et le service nous renvoie une réponse.

Cela ce présente de la façon suivante :



Le modèle nous envoie ensuite en réponse un score et un sentiment associé.

On voit ici que la requête est divisé en deux phrases. Une positive et une négative. Le modèle renvoie parfaitement la réponse adaptée

Avantages : - Très facile d’utilisation

- Accessible aux personnes ayant peu de connaissances techniques

Inconvénients : - Le modèle a un biais neutre

* Le service est payant au dela de 3000 requêtes/mois

**Méthode 2 : Modèle sur mesure simple**

On utilise le service Azure machine learning studio qui possède une interface de drag and drop.On assemble des blocs de code qu’on lie les uns aux autres .

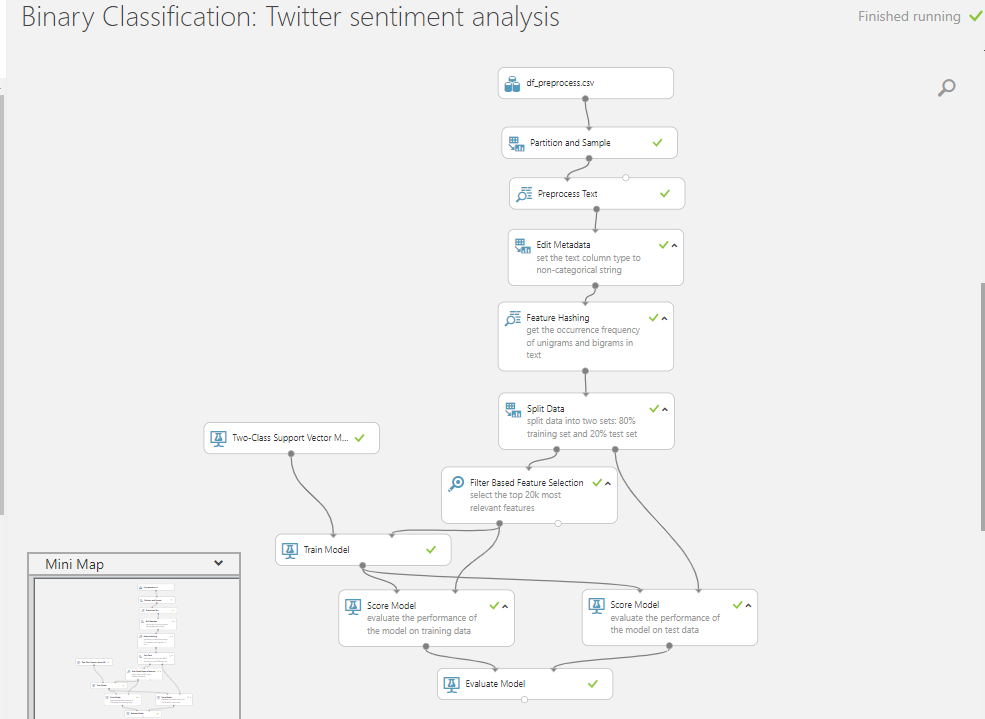
Par exemple, pour préprocess du text :

Image preprocess azure ML studio

Et voila ! on a juste a cocher les cases pour appliquer le/les filtres que l’on veut appliquer.

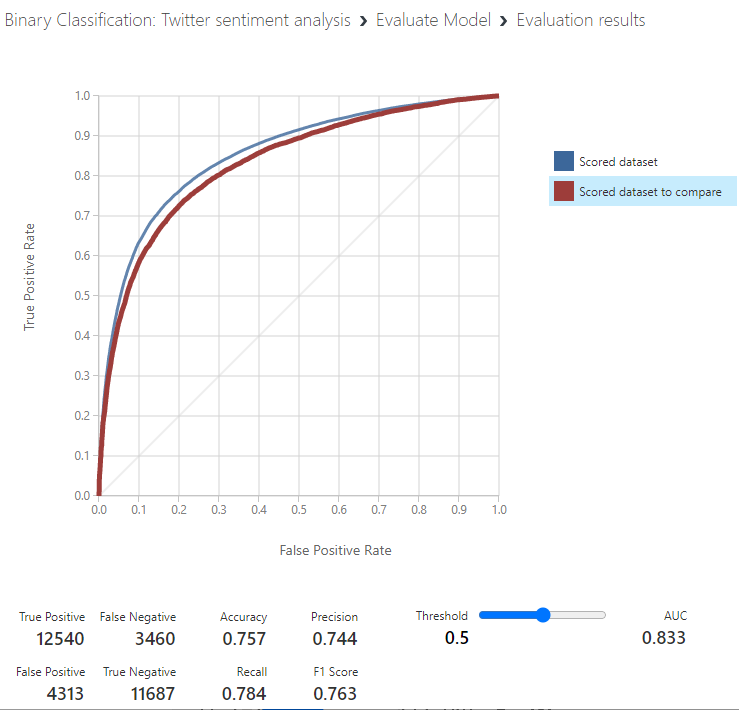
Nous allons ici créer un modèle de prédiction support vector machine learning classique.

Notre modèle final ressemblera a ceci.



Chaque bloc est lié avec un autre et en comprend vite les étapes éfféctuées . il suffit de suivre les flèches.

On peut ensuite aller dans la case evaluate model pour voir les scores de notre modèle.



0.76 de précision, voila qui n’est pas mal pour un modèle si simple !

Avantages : - Très facile d’utilisation pour les modèles simples

* Accessible aux personnes ayant peu de connaissances techniques

Inconvénients : - La prise en main est un peu difficile au début

**Méthode 3 : Modèle sur mesure avancé**

Ici on va entrainer un modèle basé sur des réseaux de neurones pour prédire le sentiment associé à un tweet

L’architecture de réseau neuronal de mémoire à long terme est populaire dans le domaine du traitement du langage naturel car elle a la capacité de conserver les informations de séquence dans sa « mémoire ».

A des fins de test, nous allons ici plusieurs techniques de word embedding (Glove et FastText) associées a deux types de réseaux de neurones différents : un CNN et un RNN

Prétraitement des données :

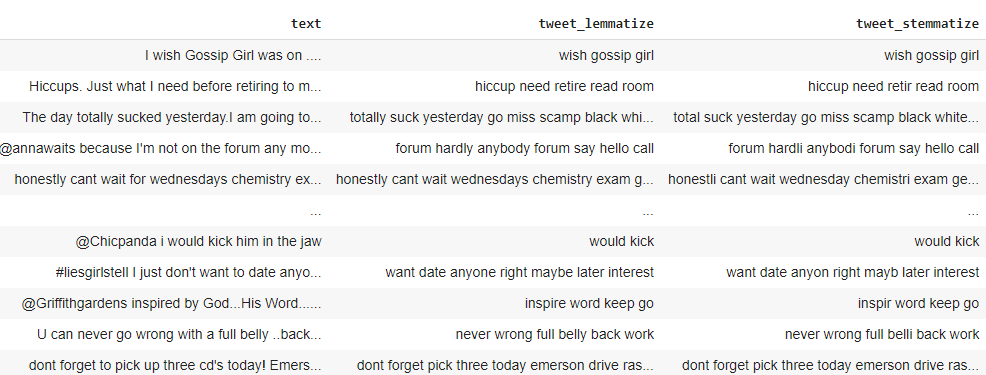
On voit que les données vont être très difficilement utilisable pour un modèle de deep learning !

Il y a beaucoup de caractères spéciaux, de liens internet, d’argot…

Il faut donc prétraiter les données afin de faciliter la tache a notre modèle.

Pour cela, nous allons donc premièrement retirer tous les stopswords, les liens internets, les caractères spéciaux, et les mots de moins de 4 lettres (qui sont trop communs et sont souvent trop présents et biaise le modèle)

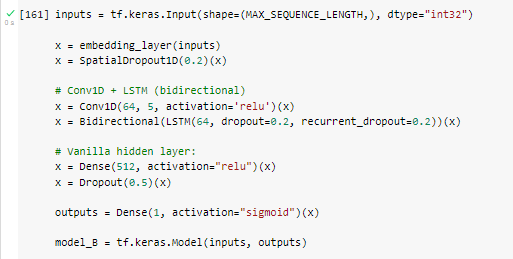
Ensuite, nous allons tokenizer les données puis appliquer une lemmatization ainsi qu’une stemmatisation pour voir lequel fonctionne le mieux.



Voila qui est beaucoup mieux pour un réseau de neurones !

Entrainement des réseaux de neurones :

Il s’agit ensuite d’entrainer les différents modèles sur ces données prétraitées.



Le code python pour programmer un reseau de neurones avec tensorflow ressemble a ceci.

De toutes les combinaisons que nous avons faites. Le RNN couplé a Glove est celui qui fonctionne le mieux sur nos données .

Déploiement dans le cloud :

Une fois notre modèle entrainé et prêt a l’emploi, il n’y a plus qu’a le déployer dans le cloud pour l’appeler via une API !

Pour ce faire, on utilise le service Azure Machine Learning qui permet d’obtenir un point REST pour notre réseau de neurones afin de lui envoyer une requête.

On aura en réponse un score qui correspond a la probabilité donnée par notre modèle ainsi qu’un sentiment associé.

Avantages : - Modèle plus précis

- Modèle adapté au problème

Désavantages : - Tout coder est compliqué sans connaissances techniques

- Déployer un modèle sur azure est assez complexe la première fois