Prérequis :

Création d’un environnement virtuel et installation de toutes les bibliothèques nécessaires

Les principales bibliothèques installées ainsi que leur version sont :

* Pandas
* Numpy
* Tensorflow version 1.8.1
* Scikit-learn version 1.11.1
* Wordcloud et PIL

Etape 1 : Récupération de données par WebScrapping

Scrapping du site trustpilot à l’aide de la bibliothèque BeautifulSoup

* Constitution d’un jeu de données pour création des modèles :

Code dans GitHub sur NoteBook: « DataScientest - Projet - SatisfactionClients - 1 - WebScrapping.ipynb »

* + Scrapping de 120 000 commentaires cdiscount avec leur note sur le site truspilot
  + Mise en forme des données et enregistrement du jeu de données pour création du modèle en local dans un fichier cdiscount.csv
* Constitution d’un jeu de test séparé qui permettra de valider notre modèle sur un jeu de donnée indépendant.

Code dans GitHub sur NoteBook: « DataScientest - Projet - SatisfactionClients - 1 – WebScrapping\_datatest.ipynb »

* + Scrapping de 100 commentaires amazon sur le site Trustpilot avec une répartition des notes égales :
    - 20 commentaires avec la note 1
    - 20 commentaires avec la note 2
    - 20 commentaires avec la note 3
    - 20 commentaires avec la note 4
    - 20 commentaires avec la note 5
  + Mise en forme des données et enregistrement du jeu de test en local dans un fichier amazon\_test1.csv

Etape 2 : Tokenization et nettoyage des données

Code dans GitHub sur le NoteBook: « DataScientest - projet - SatisfactionClients - 2 - nltk - regex - Wordcloud.ipynb »

* Utilisation des bibliothèque NLTK et REGEX pour tokenizer et nettoyer le jeu de données
* Chargement des stop Word pour 3 langues (Anglais, Français et Espagnol) dans une variable stop\_words
* Création d’une fonction commentaire\_filtering qui :
  + Tokenize les commentaires par mots
  + Ne conserve que les mots contenant des lettres
  + Transforme tous les caractères en minuscule
  + Supprime les stopwords dans le jeu de données dans 3 langues (anglais, français, espagnol) à partir de la varibla stop\_words
* Application de la fonction commentaire\_filtering sur le jeu de données cdiscount et enregistrement dans un fichier cdiscount2.csv
* Création d’un 2ème fichier à partir de cdiscount2 qui catégorisera les notes en sentiment négatif (valeur 0) ou positif (valeur 1)
  + La répartition sera la suivante :
    - Les notes pour lesquelles les commentaires ont des notes de 1 et 2 sont transformées en 0 qui correspondra à un sentiment négatif
    - Les notes pour lesquelles les commentaires ont des notes de 4 et 5 sont transformées en 1 qui correspondra à un sentiment positif
    - Les lignes ayant une note de 3 (sentiment ambigu) sont supprimées de ce fichier
* Application de la fonction commentaire\_filtering et enregistrement de ce nouveau jeu de donnée dans le fichier cdiscount\_0\_1.csv

Etape 3 : Wordcloud

Code dans GitHub sur le NoteBook: « DataScientest - projet - SatisfactionClients - 2 - nltk - regex - Wordcloud.ipynb »

Afin d’avoir un aperçu des mots les plus utilisés par les clients pour les commentaires négatifs et les commentaires positifs, la bibliothèque Wordcloud a été utilisée

* Affichage des 100 mots négatifs les plus représentés dans les commentaires, sur un fond au format d’une étoile
* Affichage des 100 mots positifs les plus représentés dans les commentaires, également sur un fond au format d’une étoile

Etape 4 : Création des modèles de machine learning

* Vectorisation des mots avec CountVectorizer de Sklearn avec les paramètres suivants pour l’ensembles des modèles Sklearn testés :
  + CountVectorizer(min\_df=3, max\_features=5000)

Qualité des modèles :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Nom du modèle | Paramètres | Score modèle sur 5 notes | Score négatifs / positifs |
| Machine Learning | DecisionTreeClassifier | max\_depth=10 | 0.69 | 0.90 |
| Machine Learning | GradientBoostingClassifier | n\_estimators=25 | 0.64 | 0.90 |
| Machine Learning | TF\_IDF RamdomForestClassifier | n\_estimators=50 | 0.69 | 0.94 |

Etape 5 : Création des modèles de machine learning avec groupe de mots

On étudie maintenant les résultats que l’on peut obtenir grâce différents modèles de deep learning. Le nombre max de mots dans le dictionnaire sera de 5000

* Vectorisation des mots avec CountVectorizer de Sklearn pour des groupes de 1, 2 ou 3 mots avec les paramètres suivants pour l’ensembles des modèles Sklearn testés :
  + CountVectorizer(max\_features=5000, ngram\_range=[1, 2])
  + CountVectorizer(max\_features=5000, ngram\_range=[2, 2])
  + CountVectorizer(max\_features=5000, ngram\_range=[2, 3])

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Nom du modèle | Paramètres | Score modèle sur 5 notes | Score négatifs / positifs |
| Machine Learning | RandomForestClassifier ngram[1, 2] | n\_estimators=50 | 0.68 | 0.94 |
| Machine Learning | RandomForestClassifier ngram[2, 2] | n\_estimators=50 | 0.64 | 0.91 |
| Machine Learning | RandomForestClassifier ngram[3, 2] | n\_estimators=50 | 0.63 | 0.92 |

Etape 6 : Création des modèles de deep learning avec tensorflow

On étudie maintenant les résultats que l’on peut obtenir grâce différents modèles de deep learning. Le nombre max de mots dans le dictionnaire sera de 5000

Après étude de la distribution du nombre de mots dans chacun des commentaires, on peut choisir une valeur cohérente avec :

* 200 mots max retenus par commentaire pour l’entrainement du modèle sur les notes découpées en 5 niveaux
* 130 mots max retenus par commentaire pour l’entrainement du modèle sur les notes découpées en 2 niveaux

Pour la vectorisation des mots on utilisera la fonction texts\_to\_sequences puis pad\_sequences de keras pour convertir la chaine de vecteur sous forme de matrice avec en nombre de colonne, le nombre maximum de mot

Pour le modèle de deep learning, c’est Embedding qui est utilisé les résultats sont les suivants :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Nom du modèle | paramètres | Score 2 du modèle sur 5 notes | Score pour sentiments négatif/positif |
| Deep Learning | Embedding1 | Embedding, globalAveragePooling, Dense(64) | 0.71 | 0.95 |
| Deep Learning | Embedding2 | Embedding, globalAveragePooling, Dense(32), Dense(32) | 0.70 | 0.95 |
| Deep Learning | Embedding3 | Embedding, globalAveragePooling, Dense(256), Dense(128), Dense(64) | 0.71 | 0.96 |
| Deep Learning | Embedding4 | Embedding, LSTM(200) | 0.70 | 0.95 |
| Deep Learning | Embedding5 | Embedding, globalAveragePooling, Dense(256), Dense(128), Dense(64), Dense(32) | 0.71 | 0.95 |
| Deep Learning | Embedding6 | Embedding, globalAveragePooling, Dense(256) | 0.70 | 0.96 |
| Deep Learning | Embedding7 | Embedding, globalAveragePooling, Dense(1024) | 0.71 | 0.96 |

Etape 7 : Création dans Streamlit d’une interface web pour :

* Page 1
  + La documentation du projet
  + Présentation des données utilisées pour la construction des modèles
  + Possibilité de télécharger les données
* Page 2 :
  + Saisie du commentaire qui sera prédit
  + Choix de la méthode à utiliser
    - Machine Learning
    - Deep Learning
  + Choix du modèle utilisé pour la prédiction
  + Chargement du modèle sélectionné (entrainé auparavant et juste loader dans Streamlit)
  + Vectorisation du commentaire saisie dans streamlit
  + Application du modèle à ce commentaire
  + Affichage du score du modèle
  + Affichage de la prédiction :
    - Prédiction négative ou positive
    - Prédiction de la note entre 1 et 5
  + Question à l’utilisateur si « le résultat vous semble cohérent : oui / non »
    - Enregistrement de la réponse pour affichage dans un graphique au fur et à mesure