Etape 1 : Récupération des données

* Webscrapping du site trustpilot
  + Jeu pour création du modèle : Scrapping de 120 000 commentaires cdiscount avec leur note
  + Jeu de test : Scrapping de 100 commentaires amazon avec leur note
* Mise en forme des données et enregistrement du jeu pour création du modèle en local dans un fichier cdiscount.csv
* 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

* Utilisation des bibliothèque NLTK et REGEX pour tokenizer et nettoyer
* Chargement des stop Word
* Création d’une fonction qui :
  + Tokenize les commentaires en mots
  + Ne conserve que les mots contenant des lettres
  + Supprime les stopwords dans 3 langues (anglais, français, espagnol)
* Application de la fonction et enregistrement dans un fichier cdiscount\_retraite.csv

Etape 3 : Wordcloud

* Affichage des 100 mots les plus représentés dans les commentaires sur un fond au format d’une étoile
* Affichage des 100 mots les plus représentés dans les titres sur un fond au format d’une étoile
* Séparation des commentaires en 2 catégories :
  + Commentaires positifs avec notes de 4 et 5
    - Affichage des 100 mots les plus représentés dans les commentaires positifs sur un fond au format d’une étoile
  + Commentaires négatifs avec notes de 1 et 2
    - Affichage des 100 mots les plus représentés dans les commentaires négatifs sur un fond au format d’une étoile
* Séparation des titres en 2 catégories :
  + Titres positifs avec notes de 4 et 5
    - Affichage des 100 mots les plus représentés dans les titres positifs sur un fond au format d’une étoile
  + Titres négatifs avec notes de 1 et 2
    - Affichage des 100 mots les plus représentés dans les titres positifs 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=12000)
* Création d’un modèle de machine learning BalanceRandonForestClassifier testé avec des paramètres différents
  + n\_estimators=10, max\_depth=10 accuracy = 0.39
  + n\_estimators=10, max\_depth=100 accuracy = 0.50
* Création d’un modèle de machine learning RandonForestClassifier testé avec des paramètres différents
  + n\_estimators=10, max\_depth=10 accuracy = 0.65
  + n\_estimators=10, max\_depth=100 accuracy = 0.69
  + Paramètres par défaut accuracy = 0.69
* Création d’un modèle de machine learning DecisionTreeClassifier testé avec des paramètres différents
  + max\_depth=1 accuracy = 0.64
  + max\_depth=10 accuracy = 0.67
* Création d’un modèle de machine learning GradentBoostingClassifier testé avec des paramètres différents
  + n\_estimators=2, learning\_rate=1, random\_state=0 accuracy = 0.65
  + n\_estimators=5, learning\_rate=0.4 accuracy = 0.67
* Création d’un modèle de machine learning SVC testé avec des paramètres différents
  + C=10 accuracy = 1
  + C=100 accuracy = 1
  + C=200 accuracy = 1
* Vectorisation des mots avec TfidfVectorizer de Sklearn avec les paramètres suivants pour l’ensembles des modèles Sklearn testés :
  + TfidfVectorizer (min\_df=3)
* Création d’un modèle de machine learning BalanceRandonForestClassifier testé avec des paramètres différents
  + n\_estimators=10, max\_depth=10 accuracy = 0.43
* Création d’un modèle de machine learning RandonForestClassifier testé avec des paramètres différents
  + n\_estimators=10, max\_depth=10 accuracy = 0.65
  + (n\_estimators=50 accuracy = 0.698

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 2000

* Vectorisation des mots avec CountVectorizer de Sklearn pour des groupes de 1 et 2 mots avec les paramètres suivants pour l’ensembles des modèles Sklearn testés :
  + CountVectorizer(min\_df=3, max\_features=14000, ngram\_range=[1, 2])

Création d’un modèle de machine learning RandonForestClassifier testé avec des paramètres différents

* + n\_estimators=50, accuracy = 0.69
* Vectorisation des mots avec CountVectorizer de Sklearn pour des groupes de 2 mots avec les paramètres suivants pour l’ensembles des modèles Sklearn testés :
  + CountVectorizer(min\_df=3, max\_features=14000, ngram\_range=[2, 2])

Création d’un modèle de machine learning RandonForestClassifier testé avec des paramètres différents

* + n\_estimators=50, accuracy = 0.69
* Vectorisation des mots avec CountVectorizer de Sklearn pour des groupes de 2 et 3 mots avec les paramètres suivants pour l’ensembles des modèles Sklearn testés :
  + CountVectorizer(min\_df=3, max\_features=14000, ngram\_range=[2, 3])

Création d’un modèle de machine learning RandonForestClassifier testé avec des paramètres différents

* + n\_estimators=50, accuracy = 0.65

Etape 5 : Création des modèles de deep learning avec tensorflow

Modèle 1 :

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 2000

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

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 :

* Embedding(input\_dim=vocabulary\_size, output\_dim=embedding\_size, input\_length=max\_words, embeddings\_initializer='uniform')
* GlobalAveragePooling1D()
* Dense(units=64, activation='relu')
* Dropout(rate=0.3)
* Dense(units=6, activation='softmax')

loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy']

Accuracy 0,72

La répartition des résultats sur le jeu de donnée amazon\_test1 est le suivant

| **predictions** | **1** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- |
| **données réelles 1** | 1.00 | 0.0 | 0.00 | 0.0 |
| **2** | 0.90 | 0.1 | 0.00 | 0.0 |
| **3** | 0.60 | 0.1 | 0.30 | 0.0 |
| **4** | 0.35 | 0.2 | 0.25 | 0.2 |
| **5** | 0.15 | 0.1 | 0.05 | 0.7 |

Les prédictions correctes sont supérieures à 50% pour 1 étoile(s) avec 100%

Les prédictions correctes sont supérieures à 50% pour 5 étoile(s) avec 70%

Modèle 2 :

Dictionnaire de 4000 mots

Nombre maximum de mots par commentaire : 200 mots

* Embedding(input\_dim=vocabulary\_size, output\_dim=embedding\_size, input\_length=max\_words, embeddings\_initializer='uniform')
* GlobalAveragePooling1D()
* Dense(units=256, activation='relu')
* Dropout(rate=0.3)
* Dense(units=6, activation='softmax')

loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy']

Accuracy 0,72

La répartition des résultats sur le jeu de donnée amazon\_test1 est le suivant

| **predictions** | **1** | **2** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- | --- |
| **données réelles 1** | 0.95 | 0.00 | 0.00 | 0.0 | 0.05 |
| **2** | 0.95 | 0.00 | 0.05 | 0.0 | 0.00 |
| **3** | 0.45 | 0.00 | 0.20 | 0.3 | 0.05 |
| **4** | 0.40 | 0.00 | 0.05 | 0.3 | 0.25 |
| **5** | 0.15 | 0.05 | 0.05 | 0.0 | 0.75 |

Les prédictions correctes sont supérieures à 50% pour 1 étoile(s) avec 95%

Les prédictions correctes sont supérieures à 50% pour 5 étoile(s) avec 75%

Modèle 3 :

Dictionnaire de 2000 mots

Nombre maximum de mots par commentaire : 200 mots

* Embedding(input\_dim=vocabulary\_size, output\_dim=embedding\_size, input\_length=max\_words, embeddings\_initializer='uniform')
* GlobalAveragePooling1D()
* Dense(units=256, activation='relu')
* Dropout(rate=0.3)
* Dense(units=128, activation='relu')
* Dropout(rate=0.2)
* Dense(units=64, activation='relu')
* Dropout(rate=0.2))
* Dense(units=6, activation='softmax')

loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy']

Accuracy 0,71

La répartition des résultats sur le jeu de donnée amazon\_test1 est le suivant

| **predictions** | **1** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- |
| **données réelles 1** | 1.00 | 0.00 | 0.00 | 0.00 |
| **2** | 0.85 | 0.05 | 0.00 | 0.10 |
| **3** | 0.55 | 0.30 | 0.15 | 0.00 |
| **4** | 0.35 | 0.15 | 0.25 | 0.25 |
| **5** | 0.25 | 0.10 | 0.00 | 0.65 |

Les prédictions correctes sont supérieures à 50% pour 1 étoile(s) avec 100%

Les prédictions correctes sont supérieures à 50% pour 5 étoile(s) avec 65%

Modèle 4 :

Dictionnaire de 4000 mots

Nombre maximum de mots par commentaire : 200 mots

* Embedding(input\_dim=vocabulary\_size, output\_dim=embedding\_size, input\_length=max\_words, embeddings\_initializer='uniform')
* GlobalAveragePooling1D()
* Dense(units=64, activation='relu')
* Dropout(rate=0.2)
* Dense(units=32, activation='relu')
* Dropout(rate=0.3)
* Dense(units=6, activation='softmax')

loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy']

Accuracy 0,72

La répartition des résultats sur le jeu de donnée amazon\_test1 est le suivant

| **predictions** | **1** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- |
| **données réelles 1** | 0.90 | 0.05 | 0.05 | 0.00 |
| **2** | 0.85 | 0.15 | 0.00 | 0.00 |
| **3** | 0.40 | 0.30 | 0.15 | 0.15 |
| **4** | 0.35 | 0.15 | 0.20 | 0.30 |
| **5** | 0.15 | 0.10 | 0.00 | 0.75 |

Les prédictions correctes sont supérieures à 50% pour 1 étoile(s) avec 90%

Les prédictions correctes sont supérieures à 50% pour 5 étoile(s) avec 75%

Modèle 5 : avec une couche LSTM

Dictionnaire de 4000 mots

Nombre maximum de mots par commentaire : 200 mots

* Embedding(input\_dim=vocabulary\_size, output\_dim=embedding\_size, input\_length=max\_words, embeddings\_initializer='uniform')
* LSTM(200)
* Dense(units=6, activation='softmax')

loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy']

Accuracy 0,71

La répartition des résultats sur le jeu de donnée amazon\_test1 est le suivant

| **predictions** | **1** | **3** | **4** | **5** |
| --- | --- | --- | --- | --- |
| **données réelles 1** | 1.00 | 0.00 | 0.00 | 0.00 |
| **2** | 0.90 | 0.00 | 0.00 | 0.10 |
|  | 0.65 | 0.10 | 0.15 | 0.10 |
| **4** | 0.50 | 0.05 | 0.05 | 0.40 |
| **5** | 0.45 | 0.00 | 0.00 | 0.55 |

Les prédictions correctes sont supérieures à 50% pour 1 étoile(s) avec 100%

Les prédictions correctes sont supérieures à 50% pour 5 étoile(s) avec 55%

Etape 6 : Création dans Streamlit d’une interface web pour :

* La documentation du projet
* l’utilisation des modèles sur des commentaires rentrés par un utilisateur
* l’affichage des prédictions (notre d’étoiles) pour les commentaires et le modèle sélectionné