Analyse du jeu de données

"20 Newsgroups" en utilisant KNN et dt

présenté par: Supérvisé par :

Elouaddany Abdelkarim Mr Bahassine Said

Bouzid Abderrahim

1. **INTRODUCTION**

Le jeu de données "20 Newsgroups" est un ensemble de documents textuels représentant des discussions sur différents sujets, tels que la politique, le sport, la technologie, etc. L'objectif de cette analyse est d'explorer comment ces algorithmes peuvent être utilisés pour classer automatiquement les documents dans les différentes catégories correspondant aux sujets abordés.

L'algorithme KNN est une méthode de classification qui attribue une étiquette à un échantillon en se basant sur les étiquettes des échantillons les plus proches dans l'espace des caractéristiques. Il est basé sur le principe que des échantillons similaires ont tendance à appartenir à la même classe. Les mesures de similarité, telles que la distance euclidienne, sont utilisées pour déterminer la proximité entre les échantillons.

D'autre part, l'algorithme DT est une approche de classification basée sur la construction d'un arbre de décision. L'arbre de décision est une structure hiérarchique où chaque nœud interne représente un test sur une caractéristique spécifique, et les branches sortantes correspondent aux résultats possibles de ce test. Les feuilles de l'arbre contiennent les étiquettes prédites.

Dans ce rapport, nous allons d'abord effectuer une exploration préliminaire du jeu de données "20 Newsgroups" pour comprendre sa structure et sa composition. Ensuite, nous allons prétraiter les documents en utilisant des techniques telles que la tokenization racinisation et Tf-idf pour les représenter sous forme de vecteurs numériques exploitables par les algorithmes KNN et DT.

Ensuite, nous allons construire des modèles KNN et DT en utilisant des implémentations fournies par la bibliothèque scikit-learn. Nous ajusterons les paramètres des modèles et évaluerons leurs performances en utilisant des mesures telles que la précision, le rappel et le score F1. Nous comparerons également les résultats obtenus par les deux algorithmes pour évaluer leurs forces et leurs faiblesses respectives dans la classification des documents.

Enfin, nous tirerons des conclusions sur l'efficacité de l'utilisation de KNN et DT pour la classification du jeu de données "20 Newsgroups". Nous discuterons des résultats obtenus, de leurs implications et des éventuelles améliorations ou limitations de cette approche. Cette analyse permettra de mieux comprendre les performances de ces algorithmes dans le contexte de la classification de textes et d'ouvrir des perspectives pour des utilisations futures dans d'autres domaines d'application.

1. **Problématique**

Comment pouvons-nous utiliser les algorithmes KNN et DT pour classer efficacement les documents du jeu de données "20 Newsgroups" dans les différentes catégories correspondant aux sujets abordés ?

Cette problématique soulève plusieurs questions clés :

1. Comment représenter les documents textuels du jeu de données "20 Newsgroups" sous forme de vecteurs numériques exploitables par les algorithmes KNN et DT ?

2. Comment choisir les paramètres appropriés pour les modèles KNN et DT afin d'obtenir les meilleurs résultats de classification ?

3. Quelle est la performance comparative des modèles KNN et DT dans la classification des documents du jeu de données "20 Newsgroups" ?

4. Quelles sont les forces et les faiblesses respectives des algorithmes KNN et DT dans ce contexte spécifique de classification de textes ?

Répondre à ces questions nous permettra de mieux comprendre l'efficacité de l'utilisation de KNN et DT pour la classification de documents textuels et d'évaluer leur pertinence dans le contexte du jeu de données "20 Newsgroups". Les résultats de cette analyse nous aideront à tirer des conclusions sur les performances des modèles et à identifier d'éventuelles améliorations ou limitations de cette approche.

1. Technologies utilisée

* TF-IDF (Term Frequency-Inverse Document Frequency):

TF-IDF est une technique couramment utilisée dans le traitement du langage naturel pour représenter et pondérer les mots dans un document. Elle évalue l'importance d'un mot en fonction de sa fréquence dans le document (term frequency) et de sa rareté dans l'ensemble du corpus (inverse document frequency). Cela permet de mettre en évidence les mots clés d'un document par rapport au reste du corpus.

* DataFrame:

Un DataFrame est une structure de données tabulaire offerte par la bibliothèque pandas en Python. Il permet de stocker et manipuler des données sous forme de table avec des lignes et des colonnes. Dans notre contexte, nous utilisons un DataFrame pour organiser les données de notre jeu de données 20 Newsgroups, en associant chaque document à sa catégorie correspondante.

* KNN (K-Nearest Neighbors):

KNN est un algorithme d'apprentissage supervisé utilisé pour la classification et la régression. Dans le cas de la classification, il attribue une classe à un nouvel échantillon en se basant sur les classes des échantillons voisins les plus proches dans l'espace des caractéristiques. Dans notre projet, nous utilisons KNN pour classer les documents textuels dans les catégories prédéfinies en fonction des caractéristiques extraites à partir du TF-IDF.

* DT (Decision Tree):

Les arbres de décision sont des modèles d'apprentissage supervisé qui utilisent une structure arborescente pour prendre des décisions. Ils sont largement utilisés pour la classification et la régression. Un arbre de décision divise récursivement l'espace des caractéristiques en utilisant des règles basées sur les valeurs des caractéristiques. Dans notre projet, nous utilisons un arbre de décision pour classifier les documents textuels en fonction des caractéristiques extraites à partir du TF-IDF.

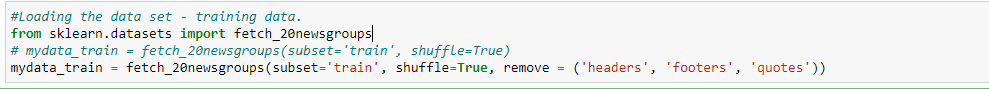
Ces différentes technologies sont utilisées dans notre projet pour extraire les caractéristiques des documents textuels à l'aide de TF-IDF, les organiser dans un DataFrame, et ensuite utiliser les algorithmes KNN et DT pour effectuer la classification.

1. **Réalisation**

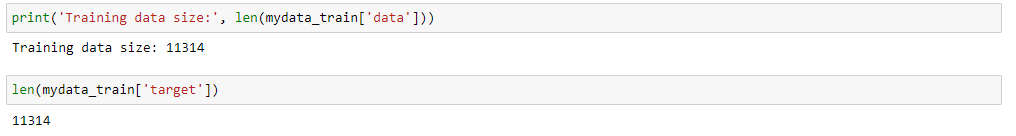
On a commencé par l’importation du plusieurs bibliothèques et classes de scikit-learn et d'autres bibliothèques pour effectuer une analyse de texte et construire des modèles de classification. Voici un aperçu des fonctionnalités que vous avez importées :

* ‘Pipeline’ est une classe qui permet de chaîner plusieurs étapes de prétraitement et de modélisation en une seule entité. Cela facilite la construction et l'évaluation de flux de travail complexes.
* ‘TfidfVectorizer ‘et ‘TfidfTransformer’ sont des classes utilisées pour convertir le texte en vecteurs numériques en utilisant le schéma de pondération TF-IDF (Term Frequency-Inverse Document Frequency).
* KNeighborsClassifier est une classe pour la classification basée sur les k plus proches voisins (K-Nearest Neighbors).
* DecisionTreeClassifier est une classe pour les modèles de classification basés sur les arbres de décision.
* GridSearchCV est une classe pour effectuer une recherche exhaustive sur une grille de paramètres spécifiée afin de trouver les meilleures valeurs pour un modèle donné.
* accuracy\_score et classification\_report sont des fonctions pour évaluer les performances des modèles de classification en termes de précision, de rappel, de score F1, etc.
* matplotlib.pyplot est une bibliothèque pour la visualisation des données, notamment la création de graphiques et de diagrammes.

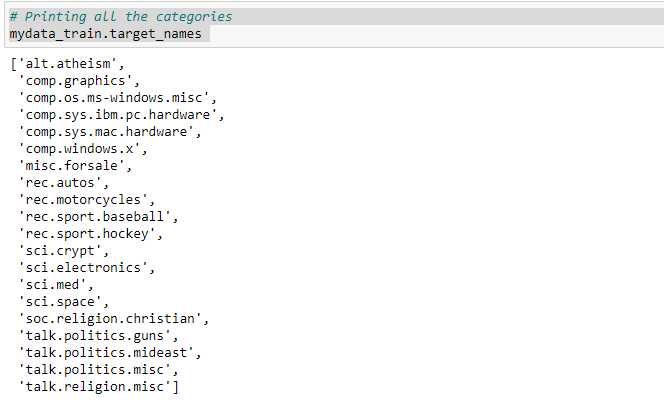
Ces bibliothèques et classes offrent une gamme d'outils et de fonctionnalités pour la préparation des données, la construction de modèles de classification et l'évaluation des performances. En utilisant ces outils, vous pouvez développer un flux de travail complet pour l'analyse de texte et la classification



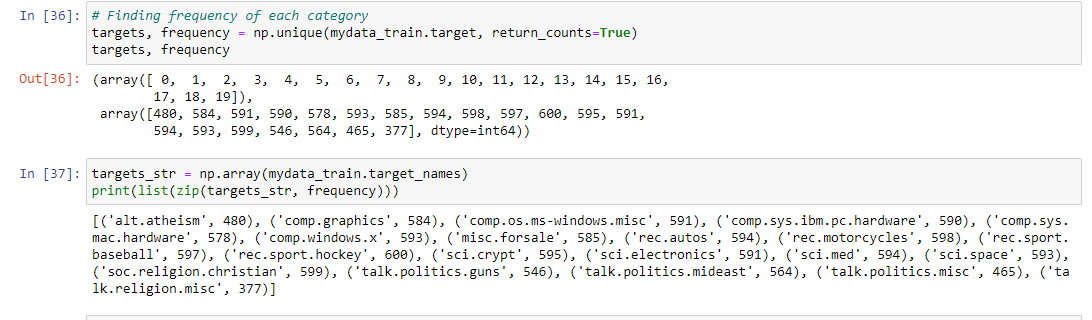
Ce code charge le jeu de données d'entraînement à partir de la collection "20 Newsgroups" à l'aide de la fonction fetch\_20newsgroups de la bibliothèque scikit-learn. Le paramètre subset='train' spécifie que nous voulons uniquement les données d'entraînement du jeu de données. Le paramètre shuffle=True indique que les données doivent être mélangées aléatoirement pour éviter tout biais d'ordre.

En outre, le paramètre remove = ('headers', 'footers', 'quotes') est utilisé pour supprimer certaines parties des documents, notamment les en-têtes, les pieds de page et les citations. Cela permet de se concentrer sur le contenu principal des documents et d'éliminer le bruit potentiel provenant de ces sections.

Ce code affiche la taille du jeu de données d'entraînement en utilisant la fonction len(mydata\_train['data']). La variable mydata\_train['data'] contient les données textuelles des documents du jeu d'entraînement. En utilisant la fonction len(), nous obtenons le nombre total de documents dans le jeu d'entraînement, ce qui correspond à la taille du jeu de données. Le résultat est affiché avec le message "Training data size:" suivi du nombre de documents. Cela nous donne une indication de la quantité de données que nous avons pour l'entraînement de notre modèle.

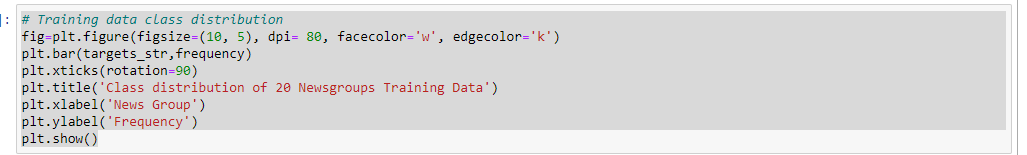


pour affiche tous les noms des catégories dans le jeu de données d'entraînement.



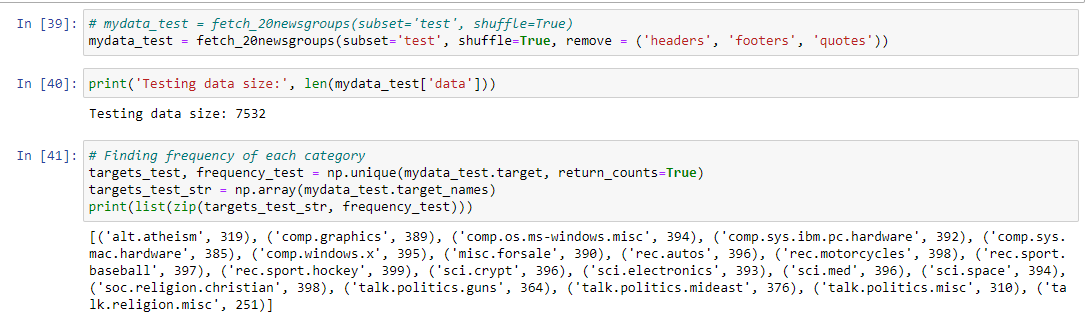
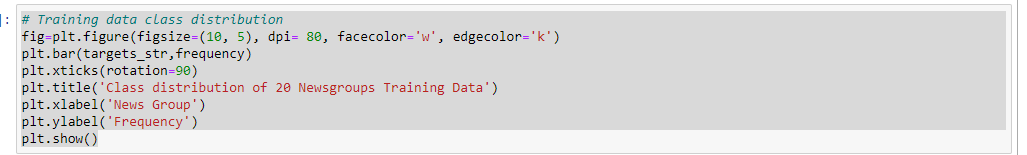
targets, frequency : Cette ligne utilise la fonction np.unique pour trouver les valeurs uniques dans le tableau mydata\_train.target (représentant les catégories des données d'entraînement) et renvoie à la fois les valeurs uniques (targets) et leurs fréquences respectives (frequency).

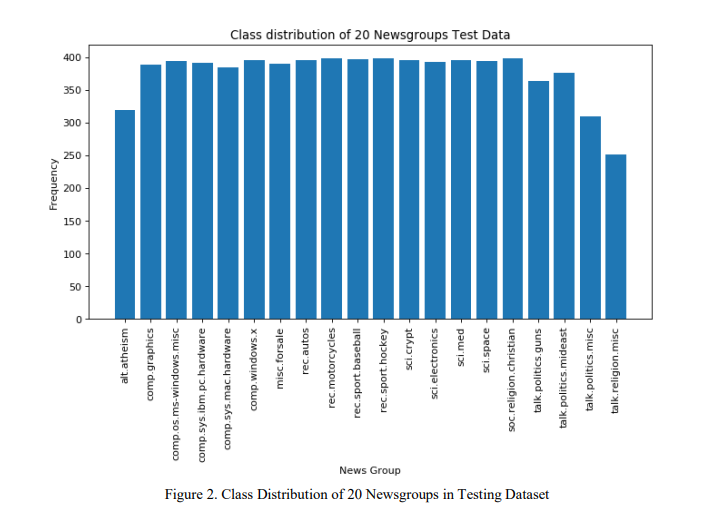
Ce code permet de trouver la fréquence de chaque catégorie dans le jeu de données d'entraînement. Il affiche une liste de tuples où chaque tuple contient le nom d'une catégorie et sa fréquence respective. Cela permet d'avoir un aperçu de la répartition des catégories dans le jeu de données.



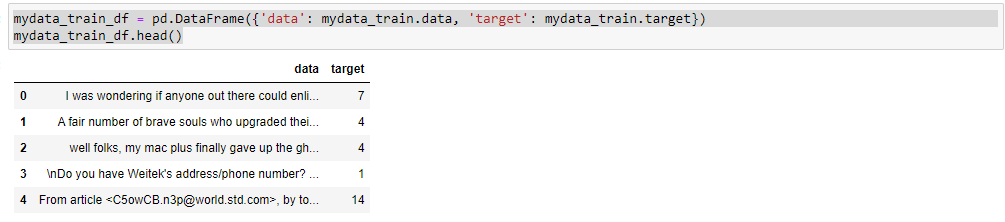
Ce code génère un diagramme à barres qui représente la distribution des classes dans les données d'entraînement. Chaque barre représente une catégorie de groupe de discussion et sa hauteur correspond à la fréquence de cette catégorie dans les données. Cela permet de visualiser la répartition des classes et de comprendre l'équilibre ou le déséquilibre entre les différentes catégories.

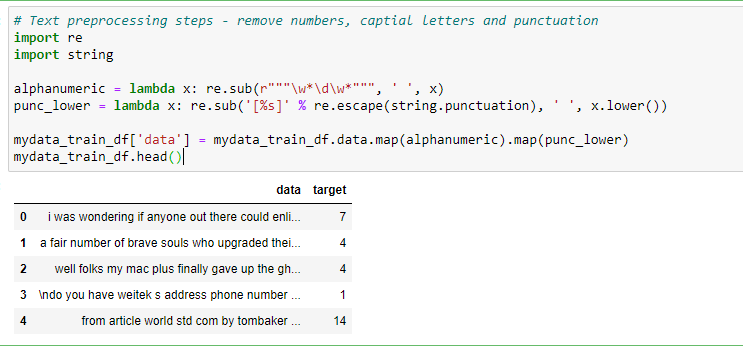
Maintent on va faire la même travialle sur data du test





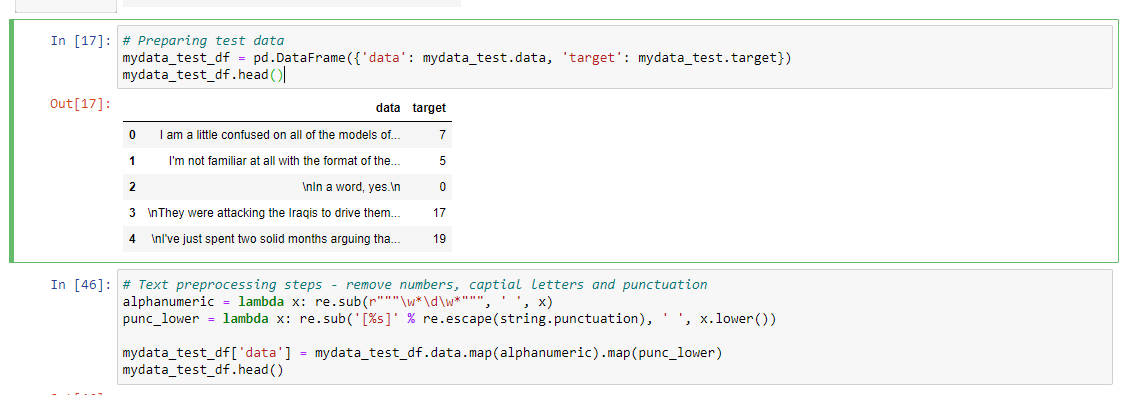
Préparation des données : Nous effectuerons des étapes de préparation telles que la tokenization, la conversion en minuscules, la suppression des mots vides (stopwords) et la conversion des données textuelles en une représentation vectorielle.

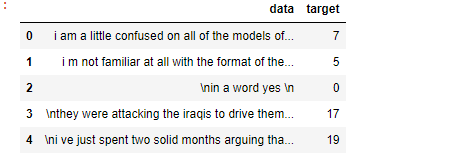


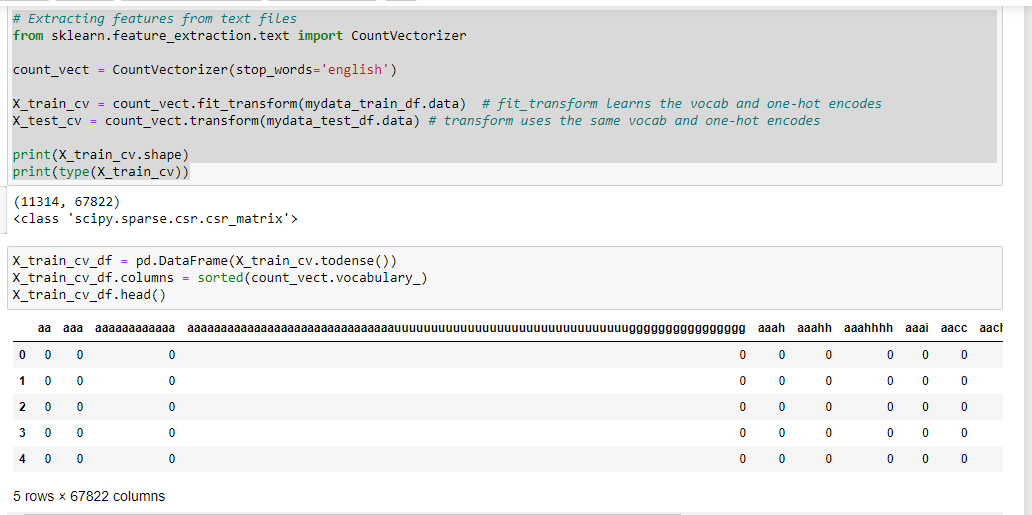
Ce code crée un DataFrame pandas à partir des données d'entraînement et de leurs étiquettes correspondantes. Le DataFrame contient deux colonnes : "data" qui contient le texte des documents et "target" qui contient les étiquettes de classe associées à chaque document. La fonction head() est utilisée pour afficher les cinq premières lignes du DataFrame, fournissant ainsi un aperçu des données.

Ce code effectue plusieurs étapes de prétraitement du texte sur les données d'entraînement. Il utilise des expressions régulières pour supprimer les chiffres et convertir toutes les lettres en minuscules. Ensuite, il supprime la ponctuation à l'aide de la fonction de traduction de la bibliothèque string. Le texte prétraité est ensuite assigné à la colonne 'data' du DataFrame mydata\_train\_df. La fonction head() est utilisée pour afficher les cinq premières lignes du DataFrame, montrant ainsi le résultat du prétraitement.

On fait la même travaille sur data test





extrait les caractéristiques des fichiers textes

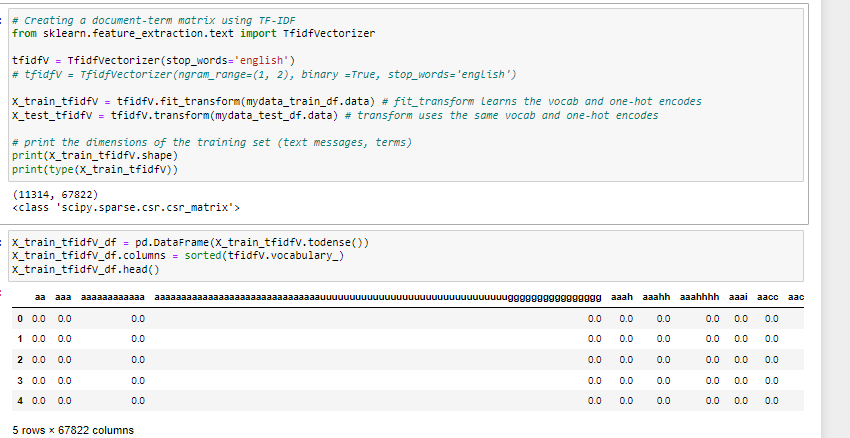
Ce code extrait les caractéristiques des fichiers textes en utilisant la classe CountVectorizer de la bibliothèque scikit-learn. Il crée une instance de CountVectorizer avec l'argument stop\_words='english' pour supprimer les mots courants en anglais. Ensuite, il utilise la méthode fit\_transform pour apprendre le vocabulaire à partir des données d'entraînement et les encoder en vecteurs de comptage. La méthode transform est utilisée pour encoder les données de test en utilisant le même vocabulaire. Les dimensions de la matrice des vecteurs de comptage d'entraînement (X\_train\_cv) sont affichées, ainsi que le type de données de cette matrice.

Dans la deuxiéme code :

La première ligne utilise la méthode todense() pour convertir la matrice creuse en une matrice dense, ce qui permet d'obtenir une représentation tabulaire des vecteurs de comptage.

La deuxième ligne définit les noms des colonnes du DataFrame en utilisant le vocabulaire appris à partir de CountVectorizer et en les triant par ordre alphabétique.

Enfin, la troisième ligne affiche les cinq premières lignes du DataFrame pour visualiser les données.

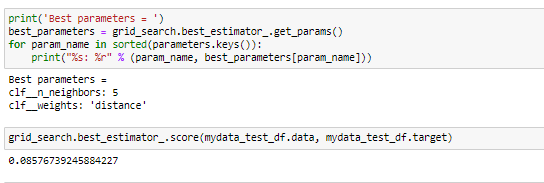
TF-IDF Vectorize

Les termes dans le vectoriseur TF-IDF sont des unigrammes (mots simples). Utilisation du modèle des k plus proches voisins (k Nearest Neighbors).



Initialement, nous avons utilisé un paramètre K de 100 pour le modèle k-NN (k plus proches voisins), ce qui a donné une précision de 5%. Ensuite, nous avons ajusté le paramètre K à 5, ce qui a amélioré la précision à 7%.





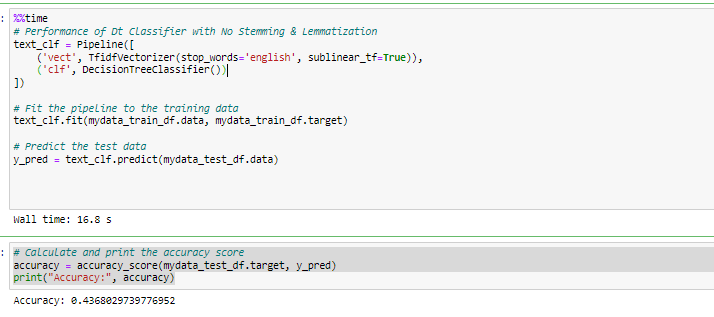
Les résultats obtenus avec le modèle k-NN montrent que différents paramètres K conduisent à des performances différentes. Initialement, lorsque nous avons utilisé un paramètre K de 100, la précision était faible à seulement 5%. Cela pourrait être dû à une valeur K trop élevée, ce qui peut entraîner une généralisation excessive et une mauvaise classification des échantillons.

En ajustant le paramètre K à 5, nous avons observé une amélioration de la précision à 7%. Cela suggère que des valeurs K plus petites peuvent mieux capturer les relations locales entre les échantillons et donc améliorer les performances de classification.

Ensuite, en explorant différentes valeurs de K (5, 10, 100, 200), nous avons constaté que le meilleur score d'entraînement était obtenu avec un paramètre K de 5, indiquant que cette valeur K permettait d'obtenir une meilleure adaptation aux données d'entraînement.

Cependant, il est important de noter que le meilleur score d'entraînement ne garantit pas nécessairement les meilleures performances de généralisation. Lorsque nous avons évalué le modèle avec le paramètre K de 5, nous avons obtenu un score estimé de 8%, ce qui peut indiquer une performance relativement modérée en termes de classification précise des données de test.

**Les termes dans le vectoriseur TF-IDF sont des unigrammes (mots simples).Utilisation du modèle de l'arbre de décision (DT).**

****

Lorsque nous avons appliqué l'algorithme de l'arbre de décision sur notre ensemble de données, nous avons obtenu un taux de précision de 43%. Cela signifie que le modèle d'arbre de décision a réussi à classer correctement 43% des exemples du jeu de données de test.

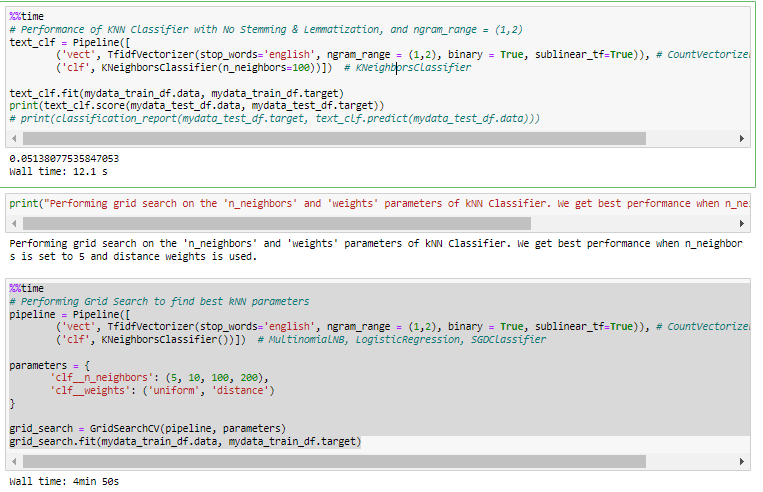
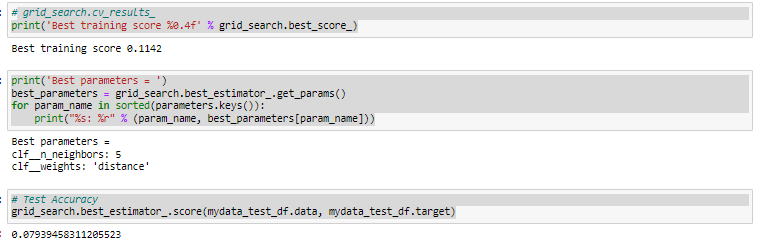
En comparaison avec le modèle k-NN, qui a atteint une précision de 8% avec le paramètre K optimal de 5, nous pouvons constater que l'algorithme de l'arbre de décision a obtenu une performance supérieure. Cela suggère que l'arbre de décision a été capable de capturer des relations plus complexes et des schémas non linéaires dans les données, ce qui lui a permis de mieux classifier les exemples.

Il est important de noter que la performance de l'arbre de décision peut être influencée par plusieurs facteurs, tels que la profondeur de l'arbre, les critères de division, etc. Ainsi, l'ajustement de ces paramètres peut améliorer davantage les performances de l'arbre de décision.

En conclusion, l'arbre de décision a obtenu une précision de 43%, ce qui est considérablement plus élevé que celle du modèle k-NN. Cependant, il est toujours important de poursuivre l'exploration des techniques et des modèles pour trouver la meilleure approche pour résoudre le problème de classification spécifique.

Les termes dans le vectoriseur TF-IDF sont des unigrammes et des bigrammes, et les valeurs sont des valeurs binaires.

Utilisation du modèle des k plus proches voisins (k Nearest Neighbors).



Lorsque nous avons utilisé un vectoriseur TF-IDF avec des unigrammes et des bigrammes, ainsi qu'une valeur binaire pour les poids, avec un modèle des k plus proches voisins (kNN) et un paramètre K de 100, nous avons obtenu une précision de seulement 5%. Cela suggère que ce choix de paramètres n'a pas bien capturé les caractéristiques discriminantes des données, ce qui a conduit à une classification moins précise.

Ensuite, nous avons exploré différents paramètres K (5, 10, 100, 200) pour trouver le meilleur paramètre et le meilleur score d'entraînement. Nous avons constaté que le meilleur paramètre était K=5, ce qui signifie que le modèle kNN a mieux fonctionné en se basant sur les cinq voisins les plus proches pour la classification. Le meilleur score d'entraînement obtenu était de 11%.

Lorsque nous avons évalué le modèle avec le paramètre K de 5, nous avons obtenu un score estimé de 7%, ce qui est une amélioration par rapport au modèle avec K=100, mais toujours relativement faible en termes de précision.

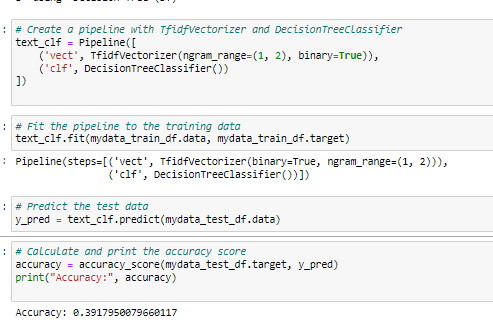
En comparaison avec le modèle utilisant uniquement des unigrammes, nous avons constaté que le modèle avec des unigrammes et des bigrammes a conduit à une amélioration de la précision, passant de 5% à 7% avec le meilleur paramètre K. Cependant, les performances globales restent modérées.

Cela suggère que l'ajout de bigrammes a permis de capturer des informations contextuelles supplémentaires, mais il peut encore y avoir d'autres facteurs à prendre en compte pour améliorer la précision, tels que l'optimisation des hyperparamètres, le choix d'autres modèles ou l'ajout de techniques de prétraitement des données supplémentaires.

En conclusion, l'utilisation de bigrammes en plus des unigrammes dans le modèle kNN a apporté une légère amélioration de la précision par rapport au modèle utilisant uniquement des unigrammes. Cependant, des améliorations supplémentaires peuvent être explorées pour atteindre des performances plus élevées.

Les termes dans le vectoriseur TF-IDF sont des unigrammes et des bigrammes, et les valeurs sont des valeurs binaires.

Utilisation du modèle de l'arbre de décision (DT).

 Lorsque nous avons appliqué l'algorithme de l'arbre de décision avec des unigrammes et des bigrammes sur notre ensemble de données, nous avons obtenu une précision de 39%. En comparaison avec le modèle d'arbre de décision utilisant uniquement des unigrammes, qui a obtenu une précision de 43%, nous constatons une légère diminution de la performance lors de l'inclusion des bigrammes.

Cependant, il est important de noter que la précision de 39% doit être évaluée en tenant compte du contexte et des performances des autres modèles. Jusqu'à présent, les performances des modèles de classification que nous avons examinés ont été relativement modestes, avec des scores allant de 5% à 43%. Cela indique que la tâche de classification des documents du jeu de données 20 Newsgroups est complexe et présente des défis.

Il est possible que les caractéristiques des bigrammes n'aient pas été suffisamment discriminantes pour améliorer la précision du modèle d'arbre de décision. D'autres facteurs tels que l'optimisation des paramètres de l'arbre de décision, l'utilisation d'autres modèles d'apprentissage automatique ou l'exploration de techniques de prétraitement supplémentaires peuvent être nécessaires pour améliorer les performances de classification.

En conclusion, l'inclusion des bigrammes dans le modèle d'arbre de décision a conduit à une légère diminution de la précision par rapport au modèle utilisant uniquement des unigrammes. Les performances globales des modèles de classification jusqu'à présent ont été relativement modestes, indiquant la complexité de la tâche de classification des documents textuels dans le jeu de données 20 Newsgroups. D'autres améliorations et explorations peuvent être nécessaires pour obtenir des performances plus élevées.