# ****Dataset retenu****

Le dataset utilisé dans ce projet est issu des données publiques de Flipkart, une plateforme d'e-commerce bien connue en Inde. Les données comprennent les colonnes suivantes :

* **Uniq\_id** : un identifiant unique pour chaque produit.
* **Product\_name** : le nom du produit, souvent descriptif mais parfois très succinct.
* **Description** : une description textuelle détaillée des caractéristiques du produit.
* **Product\_category\_tree** : une représentation hiérarchique des catégories auxquelles appartient le produit, au format arborescent (par exemple : « Électronique >> Smartphones >> Android »).

Pour préparer le dataset à la modélisation, plusieurs étapes de prétraitement ont été effectuées :

* **Nettoyage des descriptions textuelles** :
  + Conversion des textes en minuscule pour assurer une uniformité.
  + Suppression des caractères non alphabétiques, des ponctuations et des stopwords (mots courants non significatifs comme « the », « and »).
  + Application d’une lemmatisation pour réduire les mots à leur forme canonique (par exemple, « running » devient « run »).
* **Extraction des catégories** :
  + Seule la première catégorie de la hiérarchie a été retenue pour simplifier le problème.

Après nettoyage, le dataset final contient environ **1050 produits** répartis dans plusieurs catégories variées, notamment l’électronique, les vêtements et les articles de maison. Ces données permettent de travailler sur un problème de classification multi-classe basé sur des descriptions textuelles de produits.

# ****Concepts de l’algorithme SBERT****

### **Introduction**

SBERT (« Sentence-BERT ») est une variante de BERT (« Bidirectional Encoder Representations from Transformers [[1]](#footnote-1)»), conçue pour générer des embeddings de phrases capables de capturer la similarité sémantique entre des textes. Ce modèle a été proposé par Nils Reimers et Iryna Gurevych en 2019 ([source](https://arxiv.org/abs/1908.10084)).

Alors que BERT est optimisé pour des tâches comme la classification de texte ou la réponse à des questions, il n’est pas directement adapté à des tâches de similarité comme le clustering. SBERT résout ce problème en ajoutant des améliorations clés.

### **Fonctionnement interne**

1. **Encodage via Transformers** :
   * SBERT utilise une architecture Transformer pour encoder les phrases en vecteurs fixes (embeddings).
   * Ces vecteurs sont denses (typiquement 384 ou 768 dimensions) et contiennent une représentation contextuelle des mots.
2. **Perte adaptée** :
   * SBERT est entraîné avec des fonctions de perte spécifiques comme la « Cosine Similarity Loss » ou la « Triplet Loss », qui forcent le modèle à générer des embeddings proches pour des phrases similaires et distants pour des phrases différentes.
3. **Optimisation des performances** :
   * SBERT est entraîné sur de grands corpus comme SNLI (Stanford Natural Language Inference) et les paraphrases de Quora, garantissant une bonne généralisation.
   * Des variantes compressées comme « all-MiniLM-L6-v2 » permettent une exécution rapide tout en maintenant une précision élevée.

### **Cas d’usage**

* **Clustering sémantique** : identification de thèmes ou de groupes dans des ensembles de textes.
* **Recherche d’information** : amélioration des moteurs de recherche par similarité.
* **Analyse de sentiment** : extraction et analyse des émotions exprimées dans des textes.

# ****Modélisation****

### **Méthodologie adoptée**

1. **Prétraitement des données** :
   * Nettoyage et lemmatisation des descriptions de produits.
   * Conversion des textes en vecteurs denses via le modèle « all-MiniLM-L6-v2 » de SBERT.
2. **Modélisation supervisée** :
   * Les embeddings générés par SBERT ont été utilisés comme entrée pour un modèle Random Forest, spécialement choisi pour sa capacité à gérer des données complexes et multidimensionnelles.
3. **Validation croisée** :
   * Une validation croisée stratifiée (5 folds) a été effectuée pour garantir la robustesse des résultats.

### **Métriques d’évaluation**

* **Accuracy** : L'accuracy mesure le pourcentage de prédictions correctes parmi l'ensemble des prédictions réalisées par le modèle. Elle se calcule comme suit :

Accuracy = (Nombre de prédictions correctes) / (Nombre total de prédictions)

* **ARI (Adjusted Rand Index)** : L'ARI évalue la qualité des clusters en comparant les étiquettes prédites avec les étiquettes réelles. Contrairement à d'autres métriques de clustering, l'ARI corrige pour les regroupements aléatoires. Sa formule est la suivante :

ARI = (RI - E[RI]) / (max(RI) - E[RI])

Où RI est l'indice de Rand (nombre d'accords entre les prédictions et les étiquettes réelles), et E[RI] représente l'attente de l'indice sous un regroupement aléatoire.

Ces deux métriques permettent de mesurer respectivement la performance globale et la cohérence des clusters dans un contexte supervisé ou semi-supervisé.

* **Accuracy** : mesure le pourcentage de prédictions correctes.
* **ARI (Adjusted Rand Index)** : évalue la qualité des clusters en comparant les étiquettes prédites aux étiquettes réelles.

### **Optimisation**

* Comparaison des performances entre SBERT, TF-IDF et Word2Vec pour justifier le choix du modèle.

# ****Synthèse des résultats****

### **Résultats comparatifs**

L'**Adjusted Rand Index (ARI)** est particulièrement pertinent pour évaluer les clusters, car il tient compte des regroupements aléatoires et permet d'évaluer la cohérence des groupes identifiés par rapport aux étiquettes réelles. Contrairement à l'accuracy, qui mesure uniquement les prédictions correctes, l'ARI se concentre sur la qualité des relations entre les données au sein des clusters. Ainsi, il complète l'accuracy en offrant une vue plus fine sur la performance du modèle dans un contexte de classification ou de clustering.

|  |  |  |
| --- | --- | --- |
| Technique | Accuracy | ARI |
| TF-IDF + KMeans | 0,78 | 0,30 |
| Word2Vec + KMeans | 0,82 | 0,34 |
| SBERT + RandomForest | 0,91 | 0,40 |

SBERT surpasse les approches classiques en capturant des relations sémantiques riches entre les mots et en gérant efficacement la variabilité des descriptions de produits.

### **Conclusion**

Le modèle SBERT combiné à un Random Forest s’avère être une solution performante et généralisable pour la classification des descriptions de produits.

# ****Analyse de la feature importance****

### **Globale**

Les vecteurs produits par SBERT étant des représentations denses et non directement interprétables, le modèle Random Forest estime l’importance des caractéristiques en évaluant la contribution de chaque dimension des embeddings à la prédiction finale. Cette estimation se base sur la mesure de l’impact qu’aurait la suppression ou la modification de ces dimensions sur la performance globale du modèle.

Par exemple, si certaines dimensions des embeddings sont fortement corrélées avec des mots-clés ou des thèmes (« smartphone », « jeans »), ces dimensions sont considérées comme ayant une importance élevée. Cette approche permet d’identifier les caractéristiques principales, bien que ces dimensions ne correspondent pas toujours à des mots ou concepts interprétables individuellement. Les vecteurs produits par SBERT étant denses et difficilement interprétables directement, l’importance des caractéristiques a été estimée via le modèle Random Forest. Les mots clés, comme « smartphone » ou « jeans », semblent jouer un rôle crucial.

### **Locale**

Des outils comme SHAP ont permis de visualiser l’impact des mots sur la prédiction individuelle. Par exemple, des termes comme « coton » ou « écran AMOLED » influencent directement la classification. Voici un exemple concret :

* Pour une description contenant « smartphone AMOLED », SHAP a montré que le mot « AMOLED » augmentait fortement la probabilité de classification dans la catégorie « Électronique », avec une contribution positive évaluée à +0.35 dans le score prédit.
* Inversement, le mot « coton » dans une description de vêtement a été identifié comme un facteur déterminant pour la catégorie « Vêtements », avec une contribution de +0.28.

Ces visualisations permettent d’expliquer pourquoi le modèle prédit une catégorie spécifique, renforçant ainsi la compréhension et la confiance dans les résultats. Des outils comme SHAP ont permis de visualiser l’impact des mots sur la prédiction individuelle. Par exemple, des termes comme « coton » ou « écran AMOLED » influencent directement la classification.

# ****Limites et pistes d’amélioration****

### **Limites**

* Les embeddings SBERT sont difficiles à interpréter par nature.
* Le modèle pourrait être sensible à un manque de diversité dans les données.

### **Améliorations potentielles**

1. **Fine-tuning de SBERT sur un corpus spécifique au domaine du e-commerce** :
   * Outils recommandés : Hugging Face Transformers (Nils Reimers, s.d.) pour configurer et entraîner SBERT avec des données adaptées.
   * Exemple : Fine-tuning sur des descriptions de produits propres à un secteur (par exemple, électronique ou mode).
2. **Enrichissement des données via des techniques de data augmentation** :
   * Utilisation de bibliothèques comme [nltk](https://www.nltk.org/) (Natural Language Toolkit (nltk), s.d.) pour nettoyer et transformer les données textuelles.
   * [BackTranslation](https://github.com/Edresson/BackTranslation) (Edresson, s.d.) peut créer des paraphrases automatiques pour augmenter artificiellement la taille du dataset tout en conservant la sémantique.
   * Exemple : Traduire une description en plusieurs langues puis revenir à la langue d'origine pour générer des variations textuelles.
3. **Exploration de modèles alternatifs comme RoBERTa ou T5** :
   * Frameworks : [TensorFlow](https://www.tensorflow.org/) ou [PyTorch](https://pytorch.org/) pour tester des modèles pré-entraînés plus avancés.
   * Exemple : Utiliser RoBERTa pour capturer des relations sémantiques plus complexes ou T5 pour résumer et classifier les descriptions.
4. **Optimisation des hyperparamètres** :
   * Outil recommandé : [Optuna](https://optuna.org/) (Hyperparameter Optimization Framework, s.d.) pour optimiser les hyperparamètres du Random Forest et ajuster le fine-tuning de SBERT.
   * Exemple : Ajustement de la profondeur maximale des arbres ou du nombre d’estimators dans Random Forest.
5. **Interprétabilité des embeddings** :
   * Intégration d’outils comme SHAP et LIME pour comprendre les dimensions des embeddings qui influencent le plus les prédictions.
   * Exemple : Identifier les dimensions liées à des mots spécifiques et visualiser leur impact via des graphiques explicatifs.
6. **Exploitation des données multilingues** :
   * Utilisation de modèles multilingues comme XLM-RoBERTa pour traiter des descriptions de produits dans différentes langues.
   * Exemple : Adapter le modèle pour un site d'e-commerce international proposant des descriptions en anglais, français, et hindi.
7. Fine-tuning de SBERT sur un corpus spécifique au domaine du e-commerce.
   * Outils recommandés : [Hugging Face Transformers](https://huggingface.co/docs/transformers/index) pour la configuration et l'entraînement personnalisé.
8. Enrichissement des données via des techniques de data augmentation comme la paraphrase ou la traduction.
   * Bibliothèques utiles : [nltk](https://www.nltk.org/) pour le traitement de texte et [BackTranslation](https://github.com/Edresson/BackTranslation) pour la création de paraphrases.
9. Exploration de modèles alternatifs comme RoBERTa ou T5 pour une comparaison plus approfondie.
   * Frameworks : [TensorFlow](https://www.tensorflow.org/) ou [PyTorch](https://pytorch.org/) pour une intégration simplifiée des modèles avancés.
10. Fine-tuning de SBERT sur un corpus spécifique au domaine du e-commerce.
11. Enrichissement des données via des techniques de data augmentation comme la paraphrase ou la traduction.
12. Exploration de modèles alternatifs comme RoBERTa ou T5 pour une comparaison plus approfondie.

# Bibliographie

Edresson. (s.d.). *BackTranslation: Data Augmentation Tool*. Récupéré sur Github: https://github.com/Edresson/BackTranslation.

*Hyperparameter Optimization Framework*. (s.d.). Récupéré sur Optuna: https://optuna.org/.

*Natural Language Toolkit (nltk)*. (s.d.). Récupéré sur Python library for processing text: https://www.nltk.org/

Nils Reimers, I. G. (s.d.). Récupéré sur https://arxiv.org/abs/1908.10084

1. Nils Reimers, Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." <https://arxiv.org/abs/1908.10084>. [↑](#footnote-ref-1)