**Introduction**

This project focuses on developing machine learning models to categorize products based on their descriptions. The dataset includes various attributes related to products, such as names, descriptions, prices, ratings, and more. The primary objective is to build accurate models that can predict the product category from the description.

**Requirements**

To run the code, you'll need the following libraries:

* pandas for data manipulation
* numpy for numerical operations
* scikit-learn for machine learning algorithms and preprocessing
* matplotlib and seaborn for data visualization
* wordcloud for generating word clouds
* tensorflow for building and training deep learning models

Install the required packages using pip:

bash

Copy code

pip install pandas numpy scikit-learn matplotlib seaborn wordcloud tensorflow

**Data**

**Dataset Overview**

The dataset used is train\_product\_data.csv, which contains the following columns:

* **uniq\_id:** Unique identifier for each product
* **crawl\_timestamp:** Timestamp of the data scrape
* **product\_url:** URL linking to the product's page
* **product\_name:** Name of the product
* **product\_category\_tree:** Hierarchical category of the product
* **pid:** Unique product identifier on the eCommerce platform
* **retail\_price:** Original price of the product
* **discounted\_price:** Price after discounts
* **description:** Product description
* **product\_rating:** Rating given by customers
* **overall\_rating:** Aggregate rating across platforms
* **brand:** Brand name
* **product\_specifications:** Detailed specifications of the product

**Target Variable**

The target variable for classification is product\_category\_tree, which indicates the product's category.

**Preprocessing**

**Handling Missing Values**

1. **Brand Column:**
   * Missing values in the brand column are filled with the most common brand name to maintain consistency.
2. **Description Column:**
   * Rows with missing description are dropped because the description is essential for text-based feature extraction.
3. **Price Columns:**
   * Missing values in retail\_price and discounted\_price are filled with their median values to preserve data integrity.
4. **Image and Specifications Columns:**
   * Rows with missing values in image and product\_specifications are dropped, as these fields are not used in the model but could be important for other analyses.

**Text Cleaning**

The description column is cleaned to remove special characters, extra whitespace, and convert the text to lowercase. This step ensures that the text data is in a uniform format for feature extraction.

**Feature Engineering**

The description column is converted into TF-IDF features, which represent the importance of words in the descriptions relative to their frequency across all descriptions. TF-IDF (Term Frequency-Inverse Document Frequency) helps capture the relevance of words while reducing the impact of frequently occurring, less informative words.

**Label Encoding**

The product\_category\_tree is encoded into numeric labels using LabelEncoder to prepare the target variable for classification algorithms.

**Modeling**

Three different models are implemented to categorize products:

**1. Logistic Regression**

**Description:**

* Logistic Regression is a simple yet effective classification algorithm that predicts the probability of a categorical dependent variable. It works well with linearly separable data.

**Why Chosen:**

* Logistic Regression is chosen for its simplicity and efficiency in binary and multi-class classification problems. It serves as a good baseline model for comparison.

**Implementation:**

python

Copy code

from sklearn.linear\_model import LogisticRegression

lr\_model = LogisticRegression(max\_iter=1000)

lr\_model.fit(X\_train\_tfidf, y\_train)

**2. Random Forest**

**Description:**

* Random Forest is an ensemble learning method that constructs multiple decision trees and combines their outputs. It improves classification accuracy and handles both linear and non-linear data.

**Why Chosen:**

* Random Forest is selected due to its robustness and ability to handle complex interactions between features. It also helps in mitigating overfitting compared to individual decision trees.

**Implementation:**

python

Copy code

from sklearn.ensemble import RandomForestClassifier

rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_model.fit(X\_train\_tfidf, y\_train)

**3. Deep Learning**

**Description:**

* A Deep Learning model using a Sequential neural network with Dense and Dropout layers is employed. This model is capable of learning complex patterns in data through multiple layers.

**Why Chosen:**

* Deep Learning is chosen for its ability to capture intricate relationships and interactions within the data. It is particularly useful when dealing with large and complex datasets.

**Implementation:**

python

Copy code

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

dl\_model = Sequential()

dl\_model.add(Dense(512, input\_dim=X\_train\_tfidf.shape[1], activation='relu'))

dl\_model.add(Dropout(0.5))

dl\_model.add(Dense(256, activation='relu'))

dl\_model.add(Dropout(0.5))

dl\_model.add(Dense(len(label\_encoder.classes\_), activation='softmax'))

dl\_model.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

dl\_model.fit(X\_train\_tfidf.toarray(), y\_train, epochs=10, batch\_size=32, validation\_data=(X\_val\_tfidf.toarray(), y\_val))

**Evaluation**

**Metrics**

* **Accuracy:** Measures the proportion of correct predictions.
* **Classification Report:** Provides precision, recall, F1 score, and support for each class.

## Troubleshooting and Model Evaluation Issues

### Issue: F1 Score of 0 on Test Data

Despite achieving a high accuracy (0.98) on the validation set with the Deep Learning model, the F1 score on the test data dropped to 0. This discrepancy indicates potential issues that need to be addressed.