# Data Science Hackathon

- Gouthum Kharvi

**Hackathon Topic :– Ecommerce Product Categorization**

Building and Training a Multi-class Text Classifier for E-commerce Products

## Introduction:

Product categorization for e-commerce platforms like Flipkart, Amazon, and others has become increasingly vital. Accurate product categorization directly impacts a product's visibility and its likelihood of being purchased. If a product is not placed in the correct category, customers may not be able to find it, significantly reducing its chances of being sold. In my opinion, the problem of product categorization can be addressed through both **multiclass classification** and **multilabel classification** approaches. This is because a product may not necessarily belong to just one category; it can be associated with multiple categories simultaneously. This report details my observations and conclusions from training various Machine Learning and Deep Learning models to tackle the **product categorization** problem using a **multiclass classification** approach. Specifically, the report covers data **preprocessing techniques**, **feature engineering, model selection, training processes, evaluation metrics**, and the challenges encountered. The ultimate goal is to enhance the accuracy and efficiency of product categorization to improve product discoverability and sales.

## Dataset:

Dataset contains information about various products available on Flipkart, including details such as category, price, product description, product specifications, and more. It comprises 14,999 rows and 15 columns. The product categories are presented in a hierarchical tree format. Upon examining the dataset, it is evident that it contains several non-ASCII characters, including bullets and characters from languages other than English.

# Procedure & Observations:

The entire process is meticulously divided into three main parts, each documented in a dedicated Google Colab Notebook to ensure clarity and organization. These steps encompass:

1.**Exploratory Data Analysis and Data Preprocessing**: This step involves a thorough examination of the dataset to understand its structure and patterns. It includes handling missing values, removing duplicates, and transforming non-ASCII characters. Data visualization techniques are employed to gain insights into product distributions and identify any underlying trends.

2.**Machine Learning Models for Product Categorization**: This section focuses on developing and evaluating various machine learning models, such as Logistic Regression, Decision Trees, Random Forest, and XGBoost. These models are trained to accurately categorize products based on their descriptions and specifications, using techniques like feature engineering and hyperparameter tuning.

3. **Deep Learning Models for Product Categorization**: In this part, advanced deep learning models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), are implemented using PyTorch. These models are designed to capture complex patterns in the product descriptions, improving categorization accuracy. Techniques like word embeddings and sequence modeling are explored to enhance the model's performance.

# STEP 1: EXPLORATORY DATA ANALYSIS C DATA PREPROCGSING

## I) EXPLORATORY DATA ANALYSIS:

To get a fair idea about the dataset and look for parameters and lexical features that would help us improve our Machine Learning and Deep Learning Models, exploratory data analysis of the Flipkart product dataset was carried out.

## 1) Handling NaN and Duplicate values in the dataset:

Upon summarizing the dataset, it was discovered that the highest number of NaN values appeared in the "brands" column, with additional NaN values present in the "retail_price" and "discounted_price" columns. Despite these missing values, the affected rows were retained because the dataset only contains 20,000 entries. Additionally, the columns "brands," "retail_price," and "discounted_price" were deemed less significant for predicting product categories.

For the purpose of product categorization, only the "**description**" column was considered, as it contained the most relevant information. The other columns lacked sufficient details to aid in categorizing products effectively. To enhance the dataset's analysis, a potential approach could involve **"named entity recognition" (NER).** Since certain brands exclusively manufacture products within specific categories, integrating NER could enable more accurate classification by combining product descriptions with brand information.

Consequently, unnecessary columns were dropped from the dataset. Rows with NaN values in the "description" column were removed, as they could not contribute to the analysis. Additionally, duplicate entries, identified by identical product descriptions, were also eliminated to maintain data quality. This preprocessing step ensures that the dataset is refined and focused on the most informative features, setting a solid foundation for training machine learning and deep learning models for product categorization.

## 2)Visualisation of distribution of products across brands & most common words in the raw dataset :

Using the Seaborn library, a line plot was created to visualize the total number of products available for each brand on Flipkart. The plot revealed that the majority of brands had fewer than 100 products listed, with Allure Auto standing out as the brand with the highest number of products, totaling 469 entries.

Additionally, a bar graph was plotted to display the 40 most frequent words found in the product descriptions. This analysis proved highly insightful, revealing that the most common words in the descriptions were general terms related to online shopping rather than specific details about the products themselves. Common terms such as "free shipping," "delivery," "rs," "genuine," "cash," and "flipkart" were frequently used. These generic words did not provide meaningful information to distinguish one product from another and were thus deemed irrelevant for predicting product categories.

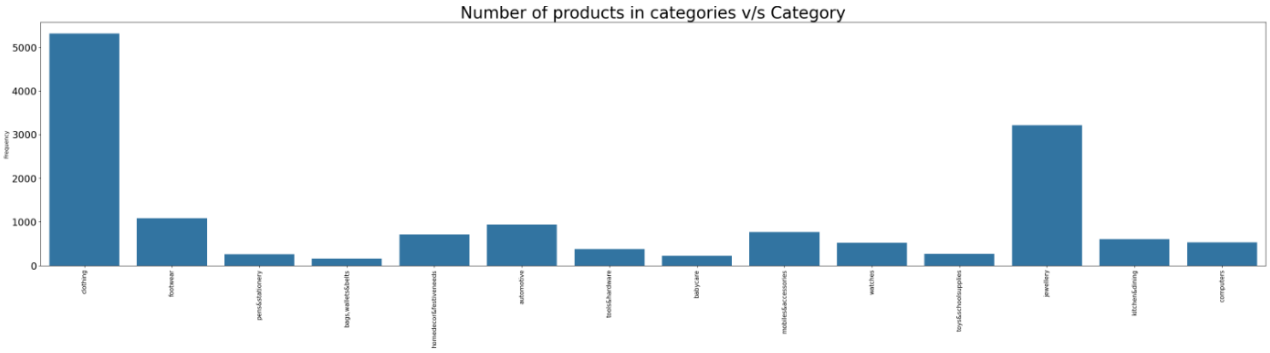
To enhance the dataset's quality and relevance, these non-informative words were removed during the data cleaning process. This step ensured that the remaining text in the product descriptions was more focused on features and details pertinent to categorization, ultimately aiding in the development of more accurate machine learning and deep learning models for product categorization.

## 3) Figuring out the Primary Categories in which a product can be classified:

On looking at the unique product categories in the dataset, it was found that most of the product category trees were made in such a way that they branched out into a particular brand. Also, this helped us realise that some data points correspond to the **noise** in the dataset as the labels (in our case, the product_category_tree column) did not specify the category but was just a mixture of the product description and the brand name. First, the labels were lowercased, and punctuation removal was performed to club the "true" categories together. It was then decided to consider **the root of the tree as the primary category for product classification**. This was done because of the following reasons:

1) With the dataset being already quite small, if any other subcategory was considered for classification, the number of entries in each category would have been quite small for some categories (example: sunglasses) while the other categories would have dominated (example: clothing and jewellery). This would have led to an even more **imbalanced dataset** which would have **compromised the accuracy** of our model on real-world data.

It was also observed that the dataset had many redundant categories with fewer entries. To **balance the dataset and increase the number of entries in a class**, these redundant categories were clubbed into a single category. This somewhat helped in balancing the number of entries in a product category of the already small dataset. On plotting a graph of the number of products in each category vs. the Categories, it was observed that the dataset was largely imbalanced with only very few columns dominating, and the rest is just noise. With the maximum number of products being around 5000 for the **clothing** category and other largely dominant categories having around hundreds of products, it was decided to **drop those categories which have less than ten products.** This was done to reduce the noise in our dataset, leading to increased accuracy of our model. After combining the redundant categories and removing the categories having less than ten products, the following was the graph obtained for the distribution of product across the categories:




**The final decided categories/ labels are 14 in number (n_classes = 14). They are:**

| | |
|---|---|
| **1. CLOTHING** | **2. Footwear** |
| **3. Pens & Stationery** | **4. Bags, Wallets & Belts** |
| **5. Home Decor & Festive Needs** | **6. Automotive** |
| **7. Tools & Hardware** | **8. Baby Care** |
| **9. Mobiles & Accessories** | **10. Watches** |
| **11. Toys & School Supplies** | **12. Jewellery** |

| | |
|---|---|
| **13. Kitchen & Dining** | **14. Computers** |

## 4) Distribution of brands across the Primary Categories and Sentiment Analysis of Product Description:

A graph was also created to display the distribution of brands across the 14 main product categories. The graph revealed that the "home furnishing/kitchen" category featured the most extensive variety of brands, with over 800 represented. In contrast, most other categories had around 400 or fewer brands.

Furthermore, Sentiment Analysis was conducted on the product descriptions by calculating the polarity scores using the TextBlob library in Python. This analysis helped determine the sentiment associated with each description. The resulting Seaborn boxplot indicated that most descriptions across all categories exhibited positive or neutral sentiments. This outcome was expected, as product descriptions typically contain factual information about the products rather than subjective reviews or opinions.

The sentiment analysis provided additional insight into the nature of the product descriptions, confirming that they predominantly conveyed neutral or positive information, which is standard for e-commerce listings. This step further enriched the dataset by adding a layer of sentiment context, which could be leveraged for more nuanced categorization or other downstream tasks.

## 5) Visualisation of the Minimum, Maximum, and Mean text length of Product Description:

A detailed analysis was conducted to examine the distribution of minimum, maximum, and average text lengths of product descriptions across all categories. This analysis aimed to determine whether a minimum and maximum threshold for description length should be established, and whether rows not meeting these criteria should be discarded.

The analysis revealed that the minimum description length was 74 characters, the maximum length was 5,309 characters, and the average length was 440 characters. The plotted graphs indicated discrepancies in the minimum description lengths across categories; however, the average and maximum description lengths were uniformly distributed.

Given that the dataset already had a relatively small number of entries for each category, it was decided not to impose any minimum or maximum threshold for description lengths. This decision was made to avoid losing potentially valuable information that could be crucial for training the machine learning and deep learning models for product categorization. By

including all text lengths, the dataset remained comprehensive, ensuring that the models could leverage as much information as possible for accurate categorization.

## 6) Word Clouds generated from the Product Description:

Word Clouds were generated to visualize the 200 most frequent words across the entire dataset and the 100 most frequent words within each product category. Creating these Word Clouds proved highly beneficial for several reasons:

1. **Custom Stopwords List Creation:** The overall Word Cloud enabled us to compile a custom list of stopwords. These stopwords, which included generic terms associated with online shopping (e.g., "free shipping," "cash delivery," "online," "India," "genuine product," "package"), were removed as they did not contribute meaningful information for product categorization. By excluding these terms, we focused on more relevant words that enhance the model's ability to accurately identify product categories.

2. **Category-Specific Insights:** The individual Word Clouds for each category provided valuable insights into the specific terms associated with each product category. This analysis helped us identify key words that are crucial for categorizing products and understand which terms should be retained in the corpus. Conversely, it also highlighted words that could be discarded due to their lack of significant lexical features, refining the dataset and improving the model's effectiveness.

Overall, the Word Clouds facilitated a more targeted approach to feature selection and data preprocessing, enhancing the model's performance in product categorization.

## II) DATA CLEANING AND PREPROCESSING:

Data cleaning and preprocessing were completed in a single comprehensive process. The cleaned, processed, and resampled datasets were then saved as CSV files for future use during model training and testing. The steps involved in data cleaning and preprocessing included:

**Handling Missing Values:** NaN values were addressed by either imputing them with suitable replacements or removing rows with critical missing data. This ensured that the dataset was complete and usable.

**Removing Duplicates:** Duplicate entries were identified and eliminated to prevent redundancy and ensure that each product description was unique.

**Text Normalization:** Text data was standardized by converting all text to lowercase and removing punctuation, special characters, and non-ASCII characters. This helped in uniform processing of the text data.

**Stopword Removal:** A custom list of stopwords was created based on the Word Cloud analysis. Generic terms that did not contribute meaningful information for product categorization were removed from the descriptions.

**Feature Selection and Engineering:** Relevant features were selected, and new features were engineered from existing data. For example, key terms identified from the Word Clouds were retained, while irrelevant words were discarded.

**Data Resampling:** To address class imbalances, the dataset was resampled using techniques such as oversampling and undersampling to ensure that each product category had a balanced number of entries.

**Saving Processed Data:** The cleaned and processed data, along with the resampled datasets, were saved in CSV format for efficient retrieval and use during subsequent model training and evaluation phases.

## 1) Contraction Mapping:

Upon examining the dataset, it became evident that it contained various elements such as emoticons, bullet points, and numerical values related to product efficiency or pricing. To assess the proportion of English language characters in the dataset, a contraction mapping analysis was performed. This analysis revealed that 82.54% of the dataset consisted solely of English characters. The remaining portion included characters from other languages, such as Chinese, as well as emoticons and bullet points.

To address these issues, a custom contraction dictionary was developed. This dictionary was designed to map specific words and symbols in the dataset to their meaningful counterparts. By doing so, the aim was to preserve valuable information and ensure that relevant terms were accurately represented in the dataset. This step was crucial for maintaining the integrity and comprehensiveness of the data, which is essential for effective machine learning and deep learning model training.

## 2) Lowercasing, removal of custom stopwords, numbers, and punctuations:

The entire product description was first converted into lowercase letters. Then a custom stopwords list was made which had **stopwords from the English language** that are available in the **nltk package**, **punctuations** available in the **string package**, and some extra words were included like **"free shipping", "rs", "cash", "delivery", "product", etc** after drawing conclusions from the previously visualized Bar plots and Word Clouds. The dataset also consisted of a lot of numbers (corresponding to the prices, product details, etc) and hence, these were also removed with the help of **Regex** as these numbers would not add much value in product categorization.

## 3) Removal of hyperlinks, extra whitespaces, and non-ASCII characters:

With the help of **Regex**, many of the hyperlinks (corresponding to the details of the product), extra whitespaces, and non-ASCII characters like emoticons, bullets, check-marks, etc were removed from the corpus.

## 4) Tokenization and Lemmatization:

Tokenization and Lemmatization were performed in a single step. **WordNetLemmatizer** available in the **nltk** package was used for Lemmatization. Initially, instead of lemmatization, stemming was used. But on further study, it was observed that it did not preserve the essence of the description and hence, did not give good results. Thus, the approach was then changed to lemmatization.

# III) BALANCING THE DATASET (RESAMPLING):

From the Dataframe below, we can clearly see that the products are distributed very non uniformly throughout the different product categories. Hence, it was essential to balance the dataset in order to achieve higher accuracy. In order to measure the performance of our model on several various kinds of datasets, the unbalanced dataset (consisting of 237 entries of **noise** data points as well) was also saved in the form of a CSV file to train and test the model.

| | primary_category | count |
|---|---|---|
| 0 | clothing | 5503 |
| 1 | jewellery | 2946 |
| 2 | homefurnishing/kitchen | 2307 |
| 3 | personalaccessories | 1578 |
| 4 | electronics | 1284 |
| 5 | footwear | 1123 |
| 6 | automotive | 1009 |
| 7 | toys&schoolsupplies | 591 |
| 8 | tools&hardware | 333 |
| 9 | babycare | 324 |
| 10 | noise | 237 |
| 11 | sports&fitness | 166 |
| 12 | petsupplies | 30 |
| 13 | ebooks | 15 |

On creating a Pandas dataframe consisting of the Primary Categories and their Frequencies, it was found out that **"clothing" (5503)** and **"jewellery" (2946)** were having the most entries with a large difference in the frequency of other categories (mostly consisting around a few hundred). Two approaches were taken while balancing the dataset, which are mentioned as follows:

**1) RANDOM OVERSAMPLING BY RESAMPLING:**

Oversampling was done randomly to increase the data points in the minority classes i.e. all the product categories except **clothing** and **jewellery**. The final samples in each category (n_samples) were then decided to be **n_samples = 3000.** This value was chosen because it was close to the frequency of the majority classes as well (so that there is not much loss of useful data). This random oversampling was implemented by the **resample** function provided by the **scikit-learn** library. The data points that referred to **"noise"** in terms of product category were also removed as it was decided that the model accuracy could be

improved by removing this noise. As this dataset was already cleaned before, the cleaned and balanced dataset was then saved as a CSV file for further use while training the ML and DL models.

### 2) RANDOM UNDERSAMPLING BY RESAMPLING:

Undersampling was done randomly to decrease the data points in the majority classes i.e. **clothing** and **jewellery**. The final samples in these categories (n_samples) were then decided to be **n_samples = 1700.** This value was chosen such that it was ensured that there is not much loss of useful information. However, there is a possibility that there might have been a loss of information for the **"clothing"** category as its data points were decreased from around 5300 to only 1700. The random undersampling was implemented by the **resample** function provided by the **scikit-learn** library. The data points that referred to **"noise"** in terms of product category were also removed as it was decided that the model accuracy could be improved by removing this noise. As this dataset was already cleaned before, the cleaned and balanced dataset was then saved as a CSV file for further use while training the ML and DL models.

## STEP 2: MACHINE LEARNING MODELS FOR PRODUCT CATEGORIZATION

Several machine learning models can be used to implement text classification methods to categorise a product into its correct category based on the **product title/description**. In this notebook, we have only used the **Product Description** as the main corpus. This is mainly because of the following two reasons:

1) On analyzing the dataset, one can notice that most of the product descriptions somehow include the name of the brand in them. Hence, there wasn't a need to specifically concatenate the brand name along with the description.

2) Furthermore, there was one limitation in trying to include the brand name along with the description. This is because there were a lot of NaN values present in the "brands" column and if we removed those data points, the already small data frame consisting of 20,000 entries would then directly shrink to around 12,700. This would have led to the loss of a lot of useful information that otherwise would have helped in increasing the accuracy of our

model. Hence, **only the product description column was considered for training the models**.

The following Machine Learning algorithms were used to train a model and tested by evaluating several metrics like classification report, confusion matrix, accuracy score, etc:

**1) Logistic Regression (both Binary and Multiclass Classification Variants)**

**2) Multinomial Naive Bayes**

**3) Linear Support Vector Machine**

**4) Decision Tree**

**5) Random Forest Classifier**

**5) K Nearest Neighbours**

Every single one of the above algorithms was tried on all 3 of the following datasets:

**1) Imbalanced Dataset:** This dataset consists of previously cleaned and preprocessed data points (with lowercasing, tokenization, lemmatization, etc). The data points mainly belonged to the 14 Primary Categories that were earlier mentioned. However, there was also some noise present in the dataset that was removed before training of the model. This was done because we thought that passing around only 237 rows that correspond to noise would not be much helpful in training a good model and might further lead to less effectiveness when it came to real-world data.

**2) Dataset Balanced using Oversampling Technique:** The imbalanced dataset was balanced using the oversampling technique (implemented using resampling). Each of the classes had around 3000 samples in them. The data points have also been cleaned and preprocessed previously and they only belong to the 14 Primary Categories that were earlier mentioned and all the noise in the dataset was removed.

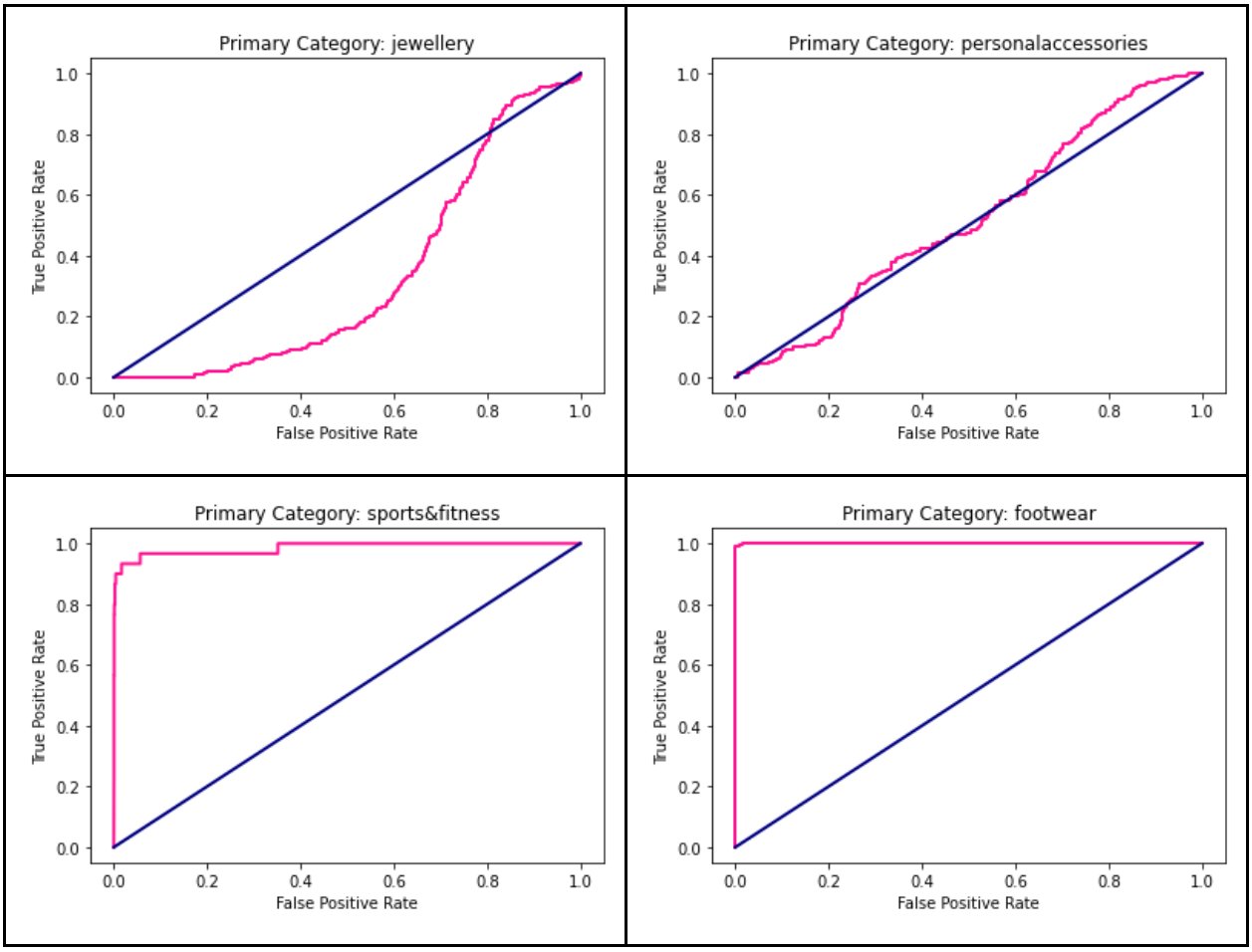
**3) Dataset Balanced using Undersampling Technique:** The imbalanced dataset was balanced using the undersampling technique (implemented using resampling). The majority classes (clothing and jewellery) were undersampled such that they only consist of 1700 samples. These data points have also been cleaned and preprocessed previously and

they only belong to the 13 Primary Categories that were earlier mentioned (all the noise in the dataset was removed).

The following table summarizes the Validation Accuracies when the models were trained and tested on the above-mentioned datasets. Other evaluation metrics like classification report and confusion matrix for all these models can be found in the second notebook.

| NAME OF THE ML ALGORITHM | IMBALANCED DATASET | BALANCED DATASET (OVERSAMPLED) | BALANCED DATASET (UNDERSAMPLED) |
|---|---|---|---|
| Logistic Regression (Binary Classification Variant) | **0.9654** | **0.9756** | **0.9486** |
| Logistic Regression (Multiclass Classification Variant) | **0.9735** | **0.9893** | **0.9654** |
| Multinomial Naive Bayes | **0.9096** | **0.9602** | **0.9054** |
| **Linear Support Vector Machine** | **0.9799** | **0.9958** | **0.9749** |
| Decision Trees | **0.7013** | **0.6883** | **0.7561** |
| Random Forest Classifier | **0.9209** | **0.9367** | **0.9235** |
| K Nearest Neighbours | **0.9564** | **0.98** | **0.9453** |

From the above table, we can clearly see that **Linear Support Vector Machine** has consistently given the best accuracy across all three datasets. From the above table, we could see that the Validation accuracy of almost all the models (except Decision Trees) is quite good. In order to further confirm this, ROC Curves were plotted and AUC Scores were calculated for the Multiclass Variant of Logistic Regression which had some interesting results. In the following table, we can see 4 out of the 13 ROC Curves that were plotted.




The ROC Curves in the first row show that for some categories, our Logistic Regression Model has a **lower True Positive Rate as compared to its False Positive Rate**. Whereas in the second row, we can see that for categories like Sports & Fitness, Footwear, etc, our model works really well. However, in reality, it is crucial for our model to work well in classifying all categories and not only a few. The AUC Score for these models was also in the range of 60-70% which is not a quite good score. Hence, we felt that **there was a need to train more advanced models by using Deep Learning for Natural Language Processing**. It was decided to move forward with only one dataset that would be used for training, validating and testing the Deep Learning models.

After a detailed analysis, it was decided to move forward with the **dataset that was balanced using the undersampling technique**. This is because of the following reasons:

1) Although the dataset that was balanced using oversampling technique has given much better accuracy as compared to the other two, we decided not to move forward with it because it was felt that this model worked really well as we randomly increased the samples for about 11 classes (all classes except clothing and jewellery). This might have led to a situation where quite similar entries were repeated in both the testing and the training dataset, thus leading to a quite biased model that has not been properly evaluated. Hence, it was decided not to proceed with the oversampled dataset as when this model might be used for real-world data, it would have worked poorly.

2) The imbalanced dataset and the dataset that was balanced using undersampling techniques more or less gave quite similar results. It was then decided to proceed with the undersampled dataset because we felt that the discrepancies in the frequency of clothing and jewellery categories were quite large to just be ignored.

Therefore, more advanced Deep Learning models like BERT and LSTM were trained and tested on the undersampled dataset.

# STEP 3: DEEP LEARNING MODELS FOR PRODUCT CATEGORIZATION

Several advanced Deep Learning-based models like Long-Short Term Memory Networks and other Transformer based models like BERT (Bidirectional Encoder Representations from Transformers) and its variants have been used for training, validation and evaluation. These models are implemented using the PyTorch framework. The models were trained and then evaluated using their accuracy score, confusion matrix and classification reports. The following are the Deep Learning models whose pre-trained models were manipulated for multiclass text classification and then evaluated:

**1) Transformer based models like:**

**a) BERT**

**b) RoBERTa (variant of BERT)**

**c) DistilBERT (variant of BERT)**

**d) XLNet (variant of BERT)**

**2) Recurrent Neural Network based LSTM**

The already cleaned and preprocessed dataset (undersampling balanced dataset) has been used for training, validation and evaluation of the Deep Learning models. Several evaluation metrics like confusion matrix, classification report, accuracy scores have been used for the comparison of the models which can be found in the third notebook. In the first part of this notebook, we have dealt with Transformer based models which is then followed by an LSTM based model. The approaches for these Deep Learning models is discussed in detail as follows:

**1) TRANSFORMER BASED MODELS: BERT**

In this part of the implementation, **BERT has been used with the huggingface PyTorch library** for efficiently fine tuning a model to achieve great accuracy in Multiclass text classification. The pre-trained model of BERT is taken in which a last layer of untrained neurons has been added. This model has then been trained to perform Product Categorisation. BERT has been chosen as a model for our NLP task because BERT has shown excellent results with **lesser amounts of data** (which closely aligns with our situation) as it has already been trained on huge chunks of information. It also has a **quicker development environment** because the pre-trained BERT model weights also contain a lot of information about English language. As compared to a more specific Deep Learning model for NLP like CNN, BiLSTM, etc, it takes a lesser **number of epochs** (recommended number of epochs is between 2 to 4) as well. Our code is mainly divided into the following three parts:

**1) BERT specific preprocessing: Tokenization and Input Formatting**

Before feeding our corpus into the BERT model, it is necessary that it is split into tokens and these tokens are then mapped to their corresponding indices. This can be performed with the help of **BertTokenizer** available in the transformers package. The BERT specific input formatting requires us to do the following:

1) Append special tokens to the start [CLS] and end [SEP] of the sentence

2) Pad and truncate all the sequences to a maximum constant length (chosen to be 128 in our case after a study of the distribution of Product Description across the corpus)

3) Explicitly differentiate the real tokens from the padded ones with **attention masks**

Our dataset is then split into training, validation and testing sets and these n-dimensional numpy arrays are converted to PyTorch tensors as well. With the help of PyTorch's DataLoader class, an iterator has also been created in order to help us save memory during execution.

**2) Training the model on our input dataset**

In the first step in training our mode, the pre-trained BERT model has been modified to give outputs for our product categorisation specific tasks. We have used **BertForSequenceClassification** for our multiclass classification. This interface is the normal pre-trained BERT model with an extra untrained layer of neurons added for performing classification related tasks.

We have chosen the AdamW optimiser for our model and the following hyperparameters for further fine-tuning:

1) **Number of epochs = 4**

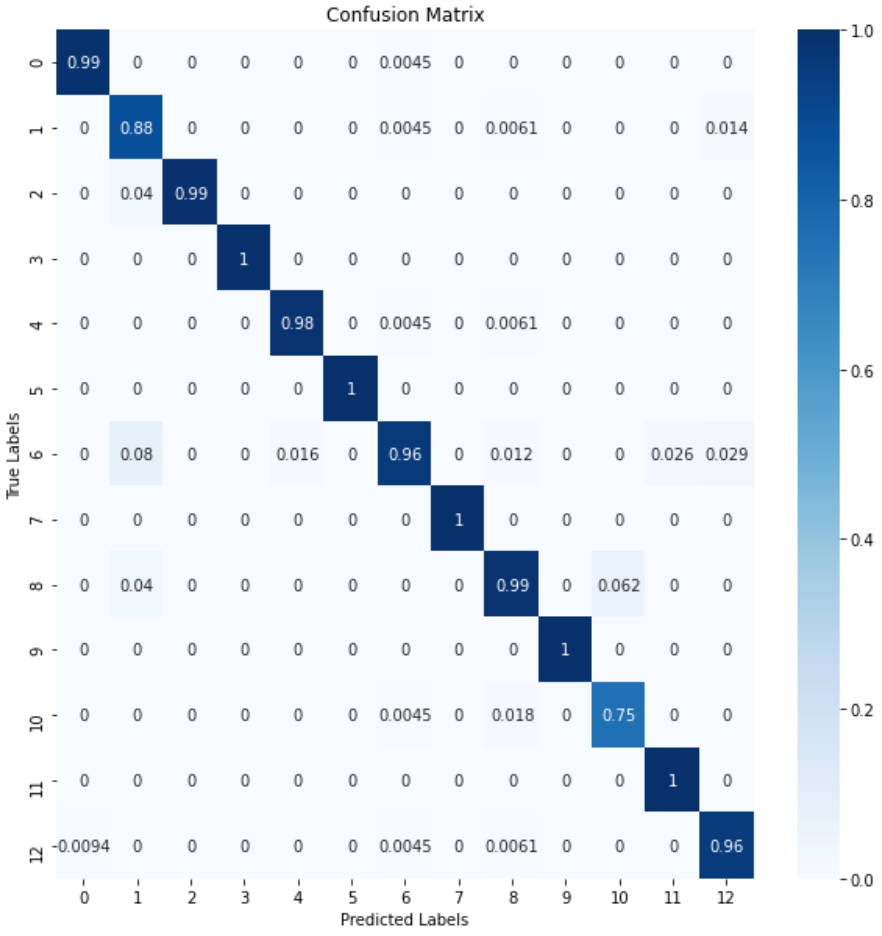
2) **Learning Rate = 3e-5**

3) **BATCH_SIZE = 32** (it is recommended to use 16 as PyTorch starts giving out of memory error while implementing other models after this)

The next step involves the main training loop in which there's a training phase and validation phase going on. In the training loop, first the model is set into the training phase. Then the description and labels are unpacked and previously calculated gradients are cleared. **Forward pass** and **backward propagation** are then executed and the model parameters are updated using the **optimizer.step()** function. Validation Accuracy and the time taken for each epoch has been calculated and can be found in the third notebook.

**3) Evaluation of the model**

To test the performance of our test set on our model, it is tokenized and formatted by following the same set of steps that were performed while preparing the training dataset. After tokenization and input formatting, an evaluation loop is run. This loop involves unpacking the description and labels and feeding the data through the network. Loss is calculated on the validation data and running variables are stored to later plot the accuracy and monitor our progress. A line graph is plotted to show the training and validation loss throughout the training of the model. Evaluation metrics like classification report and confusion matrix have been used. **This model performed well in terms of results and gave an f-1 score of 0.98.** The confusion matrix for our model can be found below:




Because of its exemplary results, finally this model has been chosen in order to perform the task of multiclass product categorisation on e-commerce product dataset.

## 2) TRANSFORMER BASED MODELS: VARIANTS OF BERT

The variants of BERT like RoBERTa, DistilBERT and XLNet were pre-trained models that were available in the Simple Transformers Package. These models were just trained by calling the **train_model()** function and evaluated using the eval_model() function. Furthermore, f1-score and accuracy scores were passed as arguments in the eval_model() function to get to know their values as well. RoBERTa pre-trained model gave an accuracy score of about **0.968**. The accuracy score for DistilBERT and XLNet couldn't be calculated because of computational limits reached on Google Colab and continuous out of limit errors posed by PyTorch because of the earlier heavy training of models.

### 3) RNN BASED LSTM NETWORK:

The process of implementation of LSTM network is also carried out in a step manner similar to that of BERT. One problem with implementing the LSTM model was that as it was the last model to be implemented, we had reached the limit of Google Colab and hence, because of computational limitations, the accuracy of this model could not be documented properly. However, the model has been created by following the steps that are discussed below:

### 1) TEXT PREPROCESSING:

Text preprocessing is done with the help of 2 different types of field objects namely **Field** and **LabelField.** The data is then split into training, validation and test sets. The next step that was performed was building the vocabulary for the product description and converting it into integer sequences to be fed into the model (initialising the words with pretrained embeddings). Each unique word in the corpus is assigned an index. **BucketIterator** was then used to create the Dataloader to further perform the tasks in batches (BATCH_SIZE was chosen to be 16).

### 2) MODEL TRAINING AND EVALUATION:

Model architecture for solving the multiclass classification problem has been defined with the help of two functions: **init()** (the constructor) and **forward()**. All the layers that are being used in the model are defined in the init constructor. The Forward function defines the forward pass of the inputs. **The** LSTM model is a variant of Recurrent Neural Network that is capable of capturing long term dependencies. Some of the parameters of this layer are input_size, hidden_size, num_layers, bi-direction (which has been set to true in our

case), etc. The optimizer is chosen to be **Adam optimizer** in this case and **CrossEntropyLoss** has been used to measure the performance. The training loop and evaluation loop respectively consist of the **Training Phase** (that sets the model in training phase and activates the dropout layers) and **Inference Phase** (that sets the model in evaluation mode and deactivates the dropout layers). The model is then run for 4 epochs and the minimum Validation Loss has been stored.