MIDAS@IIIT DELHI (INTERNSHIP SELECTION TASK)

FLIPKART PRODUCT CATEGORIZATION

- KHUSHBOO GUPTA

Introduction:

Product Categorization for e-commerce businesses like Flipkart, Amazon, etc., is becoming extremely crucial these days. This is because the probability of a product being sold is mainly dependent on product categorisation. If a product is not in its correct category, there are very high chances that a customer might not find it, and hence, it will not get sold. According to me, the problem of Product Categorization can be categorised into both, **Multiclass Classification** and **Multilabel Classification** because it is not always necessary that a product can belong to only one category. This report includes my observations and conclusions while training several Machine Learning and Deep Learning models to solve the **product categorisation problem using the Multiclass Classification approach**.

Dataset:

Link to the Dataset:
https://docs.google.com/spreadsheets/d/1on6ApK7jNXSy20Bfw_NjKMTmeICTHyBvIK-3am9irFA/edit?usp=sharing

The above dataset gives information about several products available on Flipkart and information about their category, price, product description, product specifications, etc. This dataset consists of **20000 rows and 15 columns**. The labels in the dataset i.e. the Product Category in this case was given in the form of a tree. From a look at the dataset, it can be said that there are several non-ASCII characters in the dataset as well like bullets, characters from languages other than English, etc.

Procedure & Observations:

The entire process has been divided into three parts (with a Google Colab Notebook corresponding to each component). These steps are as follows:

1) Exploratory Data Analysis and Data Pre Processing

2) Machine Learning Models for Product Categorization

3) Deep Learning Models for Product Categorization

STEP 1: EXPLORATORY DATA ANALYSIS & DATA PREPROCESSING

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:

On looking at the summary of the dataset, it was found out that the maximum number of NaN values occurred in the "**brands"** column, with some NaN values also existing in the **"retail_price"** and **"discounted_price"** columns as well. However, these rows were **not dropped** from the dataset. This is because the dataset already consists of a small number of entries (20,000 entries), and columns like **"brands," "retail_price",** and **"discounted_price"** do not contribute much in providing us with those features that can help us predict the category of a product.

Only the "description" column is considered for product categorisation as much information was not available in the other columns, which could help us categorise a product. One approach that can also be taken for an even more detailed analysis of the dataset is by doing **"named entity recognition."** As some brands exclusively make products that belong to a particular category, they could be classified into their respective categories by **combining the product description and brand**. Thus, the other unnecessary columns were dropped. Some data points had their product description as a

NaN value, and therefore, they could not be included in our dataset and were removed along with duplicate entries (entries having the same product description).

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

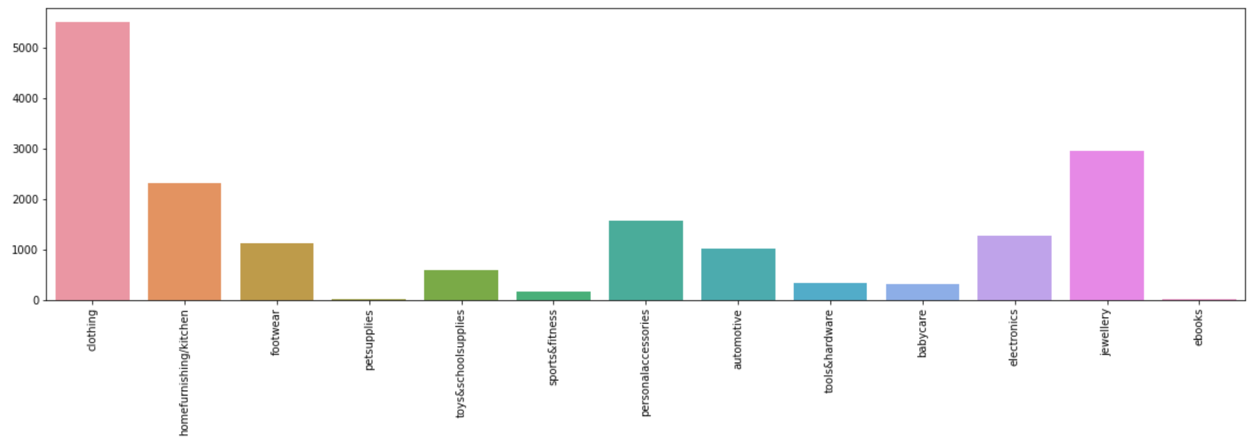
A line plot was plotted with Seaborn library's help to visualise the total number of products for each brand that was available on Flipkart. From the graph, we could see that **most of the brands had less than 100 products,** with the Allure Auto brand having the maximum products (469) entries. A bar graph was plotted showing the 40 most frequent words in the product description. Doing this was highly beneficial as it gave us an idea that **the most common words in the product description did not provide enough information about an individual product** but rather were general terms related to online shopping/ products like "free shipping", "delivery", "rs", "genuine", "cash", "flipkart", etc. As these words were not relevant features that could help predict the category of a product, they were later **removed from the description** during data cleaning.

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 13 in number (n_classes = 13). They are:

1. CLOTHING	**2. JEWELLERY**
3. HOME FURNISHING/ KITCHEN	**4. PET SUPPLIES**
5. TOYS/ SCHOOL SUPPLIES	**6. SPORTS/ FITNESS**
7. PERSONAL ACCESSORIES	**8. AUTOMOTIVE**
9. ELECTRONICS	**10. BABYCARE**
11. TOOLS/ HARDWARE	**12. EBOOKS**

13. FOOTWEAR	

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

A graph was also plotted to show the distribution of brands across the 13 main categories. From the graph, we could get to know that the category of home furnishing/ kitchen consisted of the most variety of brands, i.e., greater than 800. Most of the categories had around 400 or fewer brands. **Sentiment Analysis** on the Product Description was also performed by calculating the **polarity** with the help of the **TextBlob** library in Python. From the Seaborn boxplot, it was observed that most of the descriptions across all the categories had **positive/ neutral sentiments** associated with them. This was also expected because the product description consisted more of **factual data** describing the product and not about its reviews.

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

A detailed analysis of the distribution of minimum, maximum, and average text length of Product Description across all the categories was done. This was done to decide whether we need to keep a minimum and maximum threshold for description length and discard those rows in our dataset which do not satisfy this condition. From the graph and some calculations, it was observed that **the minimum description length was 74, the maximum description length was 5309 and the average description length was 440**. From the plotted graphs, we could see that there were **discrepancies in the minimum length** across the categories; however, the average and maximum description lengths were very **uniformly distributed**. With the dataset already being quite small with fewer entries corresponding to each class, it was decided not to keep any minimum or maximum threshold and consider all the text lengths to **prevent loss of useful information** while training the Machine Learning and Deep Learning-based models.

6) Word Clouds generated from the Product Description:

Word Clouds were created to display the 200 most frequent words in the entire dataset as well as to display the 100 most frequent words in each category. Generating Word Clouds was extremely helpful for the following reasons:

1) The major Word Cloud helped us in creating our own custom stopwords list. These stopwords were then removed as they did not add any significant value to our model by contributing to help us identify its category. These words were very generic and were mostly associated with online shopping (example: free shipping, cash delivery, online, India, genuine product, package, etc) and not any product specifically.

2) The individual Word Clouds for each category helped us in understanding what are the words the model might look for to classify a particular product in some category. It gave us an idea about what words actually matter and need to be included in the corpus and what words can be discarded as they do not have any significant lexical features.

II) DATA CLEANING AND PREPROCESSING:

Data Cleaning and Preprocessing were done once and then cleaned, processed, and resampled datasets were saved in the form of a CSV file for later use while training and testing the models. The following were the steps performed during data cleaning and preprocessing:

1) Contraction Mapping:

From one glance at the dataset, one could see that there are several emoticons, bullet points, numbers (denoting the efficiency or price of a product), etc in the dataset. Hence, contraction mapping was done to get an idea of what percentage of the dataset is comprised of only English language characters. Only 82.54% of the dataset comprised of English characters and on the further breakdown of the product description, it was found out that letters from other languages like Chinese, etc, emoticons, bullets, etc made up the rest of the dataset. Also, a custom contraction dictionary was created in order to map some of the words from the dataset to their useful meaning (those words were not simply discarded to prevent loss of information).

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:

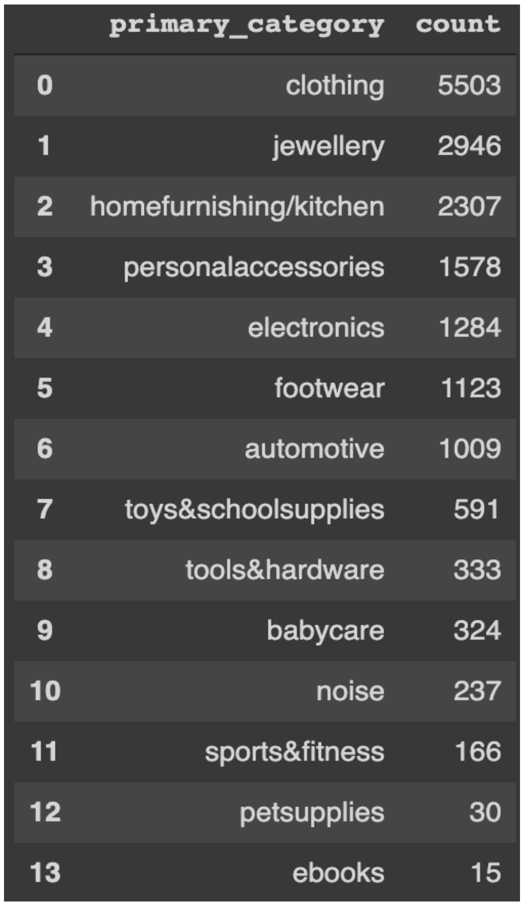
With the help of **Regex**, many of the hyperlinks (corresponding to the details of the product), extra whitespaces, and non-ASCII characters like emoticons, bullets, check-marks, etc were removed from the corpus.

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) <u>Imbalanced Dataset</u>: This dataset consists of previously cleaned and preprocessed data points (with lowercasing, tokenization, lemmatization, etc). The data points mainly belonged to the 13 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.

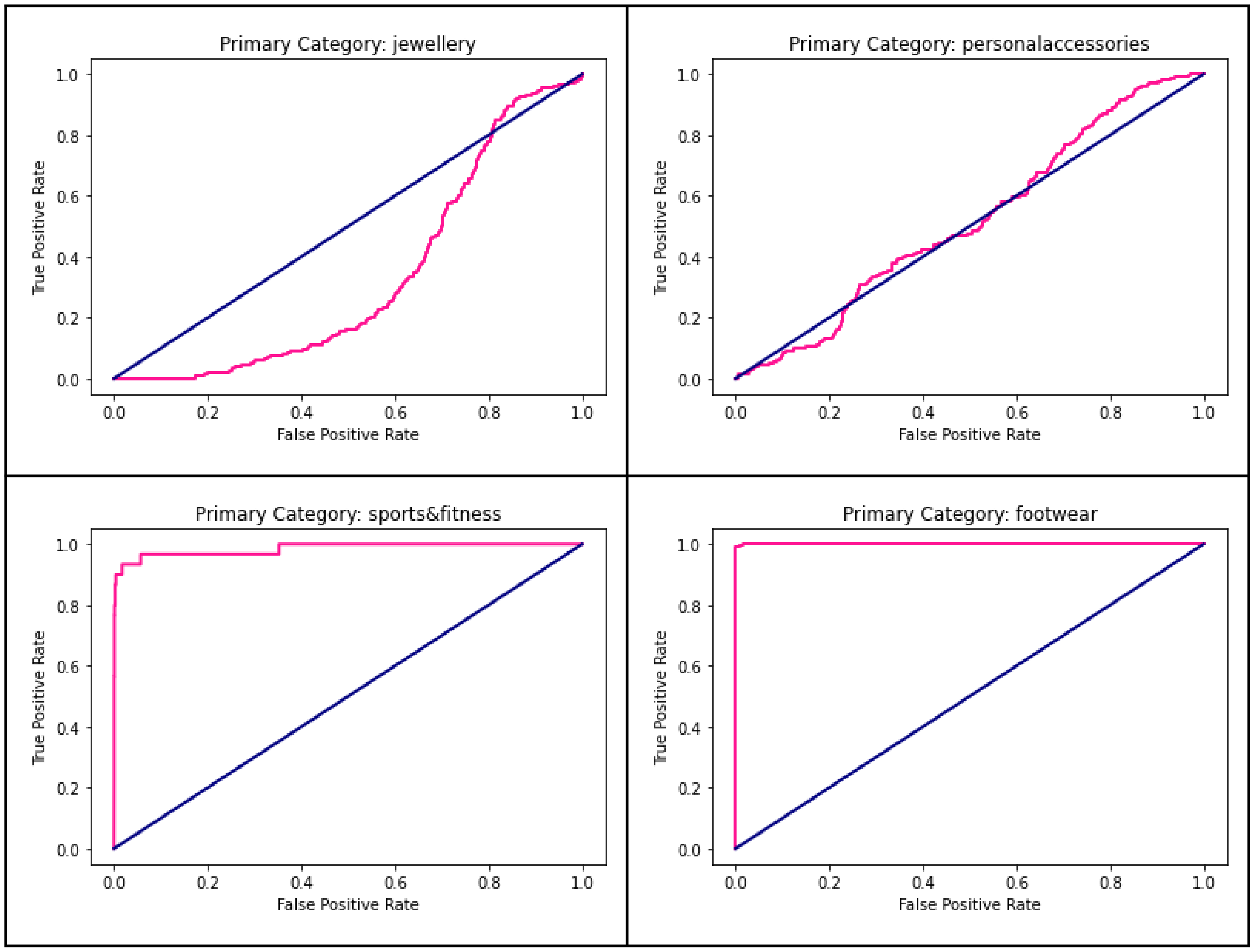
2) <u>Dataset Balanced using Oversampling Technique</u>: The imbalanced dataset was balanced using the oversampling technique (implemented using resampling). Each of the classes had around 3000 samples in them. The data points have also been cleaned and preprocessed previously and they only belong to the 13 Primary Categories that were earlier mentioned and all the noise in the dataset was removed.

3) <u>Dataset Balanced using Undersampling Technique</u>: 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**

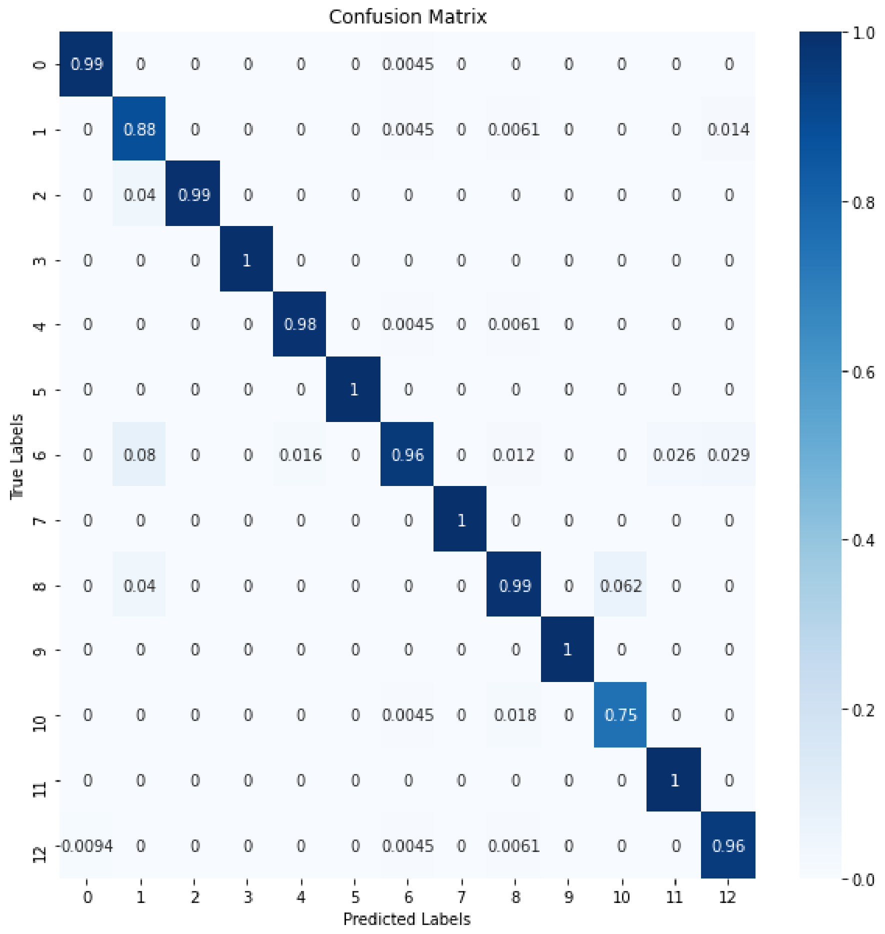
2) **Learning Rate = 3e-5**

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.

<u>2) TRANSFORMER BASED MODELS: VARIANTS OF BERT</u>

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.

<u>3) RNN BASED LSTM NETWORK:</u>

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:

<u>1) TEXT PREPROCESSING:</u>

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).

<u>2) MODEL TRAINING AND EVALUATION:</u>

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.

References:

https://arxiv.org/pdf/1905.05583.pdf

https://github.com/ThilinaRajapakse/simpletransformers#saveevalcheckpoints

https://huggingface.co/transformers/pretrained_models.html

https://towardsdatascience.com/multi-class-text-classification-with-deep-learning-using-bert-b59ca2f5c613

https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17