Flipkart Ecommerce Product Categories Prediction

Objective:

We have been given an Ecommerce Flipkart Dataset with exactly 20,000 samples. It has 15 columns with a lot of information. Our aim is to consider “Description” for a product, analyze it and predict which category it might fall under. There are few categories which are do not qualify as valid product categories. We will also see how we will handle that.

We will follow a holistic approach of step-by-step going through the data and predict product categories with highest possible true positives, thereby achieving maximum accuracy.

Approach:

The steps involved are given as follows and have been explained in detail along with outputs inside the Google Colab Notebook

1. Data Preparation: Cleaning and Analyzing data
2. Visualize this Data
3. Preprocessing textual Data
4. Model Preparation for product categorization
5. Measure the accuracy of the model
6. Can we improve the accuracy of the model?
7. What other algorithms can we try?

Requirements:

Google Colab Notebook was chosen due to two reasons:

1. RAM required (hosted on Cloud)
2. GPU required for parallel processing (for saving time)

Since the functionality remains the same as Jupyter Notebook, we would utilize the resources provided by Colab platform.

The libraries used are generic and are as follows:

1. **Pandas** (Library for handling tables and help in Data manipulation and analysis)
2. **Numpy** (Library for mathematical and computational operations)
3. **Matplotlib** (Library for plotting and visualization purposes)
4. NLTK (Tool kit for natural language processing)
5. **Sklearn** (Library for direct methods of Data splitting, Label encoding, Feature Extraction, Model implementation, Metrics like Accuracy calculation, etc)
6. **Keras** (for providing python deep learning APIs and assist in Model creation)

Implementation:

1. Data Preparation:

About Dataset: (20,000 rows, 15 columns)

Range Index: 20000 entries, 0 to 19999
Data columns (total 15 columns):

#	Column Names	Non-Null Count	Dtype
0	uniq_id	20000 non-null	object
1	crawl_timestamp	20000 non-null	object
2	product_url	20000 non-null	object
3	product_name	20000 non-null	object
4	product_category_tree	20000 non-null	object
5	pid	20000 non-null	object
6	retail_price	19922 non-null	float64
7	discounted_price	19922 non-null	float64
8	image	19997 non-null	object
9	is_FK_Advantage_product	20000 non-null	bool
10	description	19998 non-null	object
11	product_rating	20000 non-null	object
12	overall_rating	20000 non-null	object
13	brand	14136 non-null	object
14	product_specifications	19986 non-null	object

dtypes: bool(1), float64(2), object(12)

Memory usage: 2.2+ MB

The only column we are concerned for being the input will be “description” after careful inspection. And we will only consider non-null values.

Data Cleaning:

For getting the Primary Product Category, we will have to clean the “product_category_tree” column. Basically, we need to extract the primary category from the given category tree.

Example:

Original Category tree:

["Clothing >> Women's Clothing >> Lingerie, Sleep & Swimwear >> Shorts >> Alisha Shorts >> Alisha Solid Women's Cycling Shorts"]

Extracted Primary Category:

“Clothing”

We are achieving this with the help of Regex pattern matching and stripping of extra characters and words across the symbol ”>>”

Data Analysis:

After getting the primary categories, we can see some brand names in top 40 categories in our dataset. These brand names belong to main categories and are unqualified to be called as product categories.

As we can observe, these are some of the categories with frequency of 4 samples or above and are actually Brand names:

- Clovia (clothing)
- Olvin (sunglasses)
- Vishudh (clothing)
- Dressberry (clothing)
- Lilliput (baby care)
- Speedwav (automotive)
- MASARA (clothing)
- Ruhi's (clothing)
- Timberlake (clothing)
- FEET (footwear)
- Urban (beauty and personal care)

These are not valid primary categories (instead, they are Brand names) and actually belong to Clothing, Sunglasses, Baby Care, Automotive, Footwear and other categories *(conclusion on the basis of careful inspection of data)*.

So, we should replace these labels with major primary category.

After manipulating our data, we observe that only Top 28 categories are valid product categories and on the basis of these, we can classify product descriptions into their respective Primary Categories.

Top 28 categories (along with their frequency of occurrence in dataset):

Clothing	6247
Jewellery	3531
Footwear	1231
Mobiles & Accessories	1099
Automotive	1019
Home Decor & Festive Needs	929
Beauty and Personal Care	714
Home Furnishing	700
Kitchen & Dining	647
Computers	578
Watches	530
Baby Care	491
Tools & Hardware	391
Toys & School Supplies	330
Pens & Stationery	313
Bags, Wallets & Belts	265
Furniture	180
Sports & Fitness	166
Cameras & Accessories	82
Home Improvement	81
Sunglasses	56
Health & Personal Care Appliances	43
Gaming	35
Pet Supplies	30
Home & Kitchen	24
Home Entertainment	19
eBooks	15
Eyewear	10

Out of a total of 20,000 samples, we have around 19756 properly labeled and categorize d data
samples.
That means, we have around 244 samples which lack proper categorization.
So, let us remove these invalid categories (brand names) from our maintained mapping dictionary for cleaning purposes.

We will now get labels for each of these categories in numerical form.
We can either do one-hot encoding using sklearn's LabelEncoder or develop a dictionar y to map labels on our own.

Notes: We will give a label of "-1" to descriptions with no valid categories.

Labels:

```
'Automotive': 0,
'Baby Care': 1,
'Bags, Wallets & Belts': 2,
'Beauty and Personal Care': 3,
'Cameras & Accessories': 4,
'Clothing': 5,
'Computers': 6,
'Eyewear': 7,
'Footwear': 8,
'Furniture': 9,
'Gaming': 10,
'Health & Personal Care Appliances': 11,
'Home & Kitchen': 12,
'Home Decor & Festive Needs': 13,
'Home Entertainment': 14,
'Home Furnishing': 15,
'Home Improvement': 16,
'Jewellery': 17,
'Kitchen & Dining': 18,
'Mobiles & Accessories': 19,
'Pens & Stationery': 20,
'Pet Supplies': 21,
'Sports & Fitness': 22,
'Sunglasses': 23,
'Tools & Hardware': 24,
'Toys & School Supplies': 25,
'Watches': 26,
'eBooks': 27
```

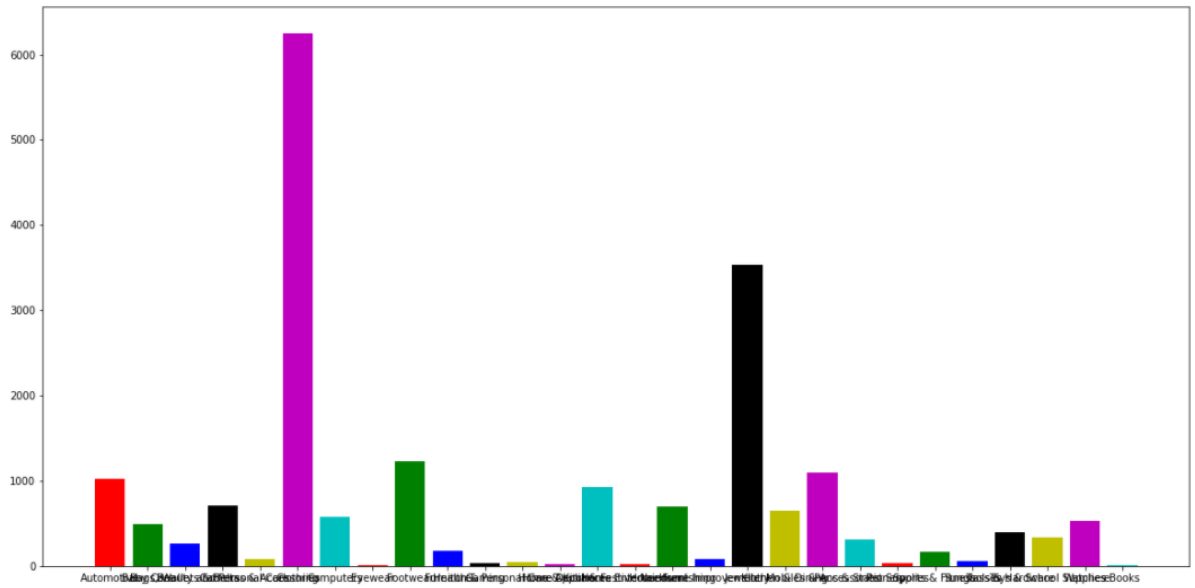
This is how our data will look now:

	description	primary_category	label
0	Key Features of Alisha Solid Women's Cycling S...	Clothing	5
1	FabHomeDecor Fabric Double Sofa Bed (Finish Co...	Furniture	9
2	Key Features of AW Bellies Sandals Wedges Heel...	Footwear	8
3	Key Features of Alisha Solid Women's Cycling S...	Clothing	5
4	Specifications of Sicons All Purpose Arnica Do...	Pet Supplies	21
5	Key Features of Eternal Gandhi Super Series Cr...	Eternal	-1
6	Key Features of Alisha Solid Women's Cycling S...	Clothing	5
7	FabHomeDecor Fabric Double Sofa Bed (Finish Co...	Furniture	9
8	Key Features of dilli bazaaar Bellies, Corpora...	Footwear	8
9	Key Features of Alisha Solid Women's Cycling S...	Clothing	5

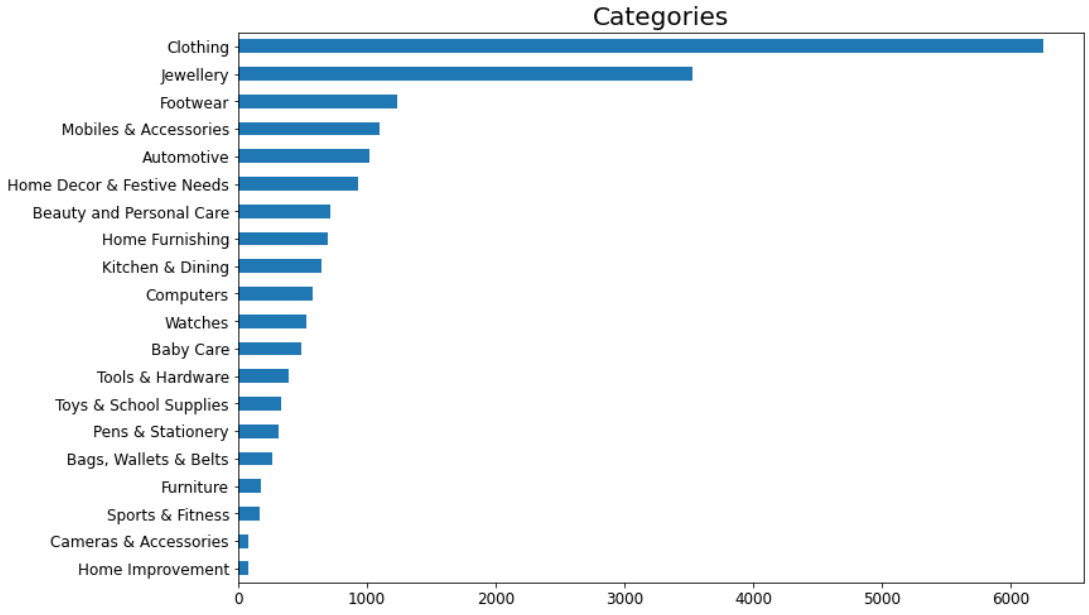
2. Data Visualization:

These graphs can be found in the notebook along with observations and deductions for the same:

All categories Frequency Distribution Bar Graph:



Top 28 categories Frequency Distribution Horizontal Bar Graph:



3. Textual Pre-Processing:

Since Classification in Machine Learning is a classic problem, we also know that classifier only understands binary (0 or 1).

Classifier does not understand languages, words, images, etc. We have to process the data in a way that our Classifier understands it in its understandable form (0 or 1).

We need to process our data (description) into a set of fixed number of features (numeric form). For this, our data will have to go through a pipeline of procedures/steps.

In our project, we will follow **Bag of Words pipeline** and get help from **NLTK** (Tool Kit) for natural language processing.

TEXT → NUMBERS → CLASSIFIER → PREDICTIONS

We will perform few steps to get clean text:

1. Removing special characters
   (%, #, @, !, &, etc)
2. Removing Stopwords
   (a, the, was, I, an, etc)
3. Removing empty spaces
   (“ “)
4. Lowercase conversion
   (“Clothing” → “clothing”)
5. **Lemmatization** (get words in raw form) ---
   WORDNET LEMMATIZER
6. **Stemming** (to convert different forms of same verb into a base word and reduce vocab size by keeping unique words only) ---
   PORTER STEMMER
7. **Tokenization** (to get all words for vocab preparation) ---
   WORDPUNKT TOKENIZER or KERAS TOKENIZER
8. **Vectorization** (to get vocabulary by converting sentences into vectors)--
   COUNT VECTORIZER or TFIDF VECTORIZER

We will also be experimenting with data to create features in different way:

(This experiment has been done in Naïve Bayes algorithm application -Notebook 2)

1. **Unigrams**

   (treating every word as a feature)
2. **Bigrams**

   (two consecutive words can be treated as single feature)
3. **Trigrams**

   (three consecutive words as single feature)
4. **N-grams**

   (n number of words, different combinations)
5. **TF-IDF normalization**

   (to avoid features that occur very often because they contain less info)

   Term Document Frequency associated a weight with every term.

Tf (term, document) = Term frequency * Inverse Document frequency

4. Model Preparation:

A classic Classification problem attracts Naïve Bayes algorithm usage. While Naïve Bayes gives a good result in this project, the accuracy was improved by using a deep learning approach. An LSTM model was used to get an accuracy of almost 95%.

Here's a bit about both the models used in this project.

A. Naïve Bayes:

Principle of Naive Bayes Classifier:

A Naive Bayes classifier is a probabilistic machine learning model that is used for classification task. It is based on the Bayes theorem.

Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The variable **y** is the class variable (Category), Variable **X** represent the parameters/features. (Description)

X is given as,

$$X = (x_1, x_2, x_3, ..., x_n)$$

Here x1, x2….xn represent the features. By substituting for **X** and expanding using the chain rule we get,

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, we can obtain the values for each by looking at the dataset and substitute them into the equation. For all entries in the dataset, the denominator does not change, it remains static. Therefore, the denominator can be removed, and a proportionality can be introduced.

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

In our case, the class variable(**y**) has 28 outcomes, product categories. (MULTIVARIATE CLASSIFICATION) Therefore, we need to find the class **y** with maximum probability.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Using the above function, we can obtain the Category, given the predictors (Descriptions).

Our project falls under **Multinomial Naive Bayes** type.

This is mostly used for document classification problem, that is whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

For us, our feature is Descriptions of products and we want to classify them as one product category (out of 28 classes).

B. RNN-LSTM Model:

When we talk about Deep Learning approach, we think Neural Networks, Multi-Layer Perceptron, CNN, etc. But these are stateless approaches. Ordering of words doesn't matter here. The only thing that matters is presence of particular set of words.

To process the document sentences word-by-word, we need **Sequence models** which can maintain a state (have a memory of what has been processed till now.

This feature can be found in Recurrent Neural Networks which work on the base level called RNN cell.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 300, 100)          2200000
_________________________________________________________________
spatial_dropout1d_2 (Spatial (None, 300, 100)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               80400
_________________________________________________________________
dense_2 (Dense)              (None, 28)                2828
=================================================================
Total params: 2,283,228
Trainable params: 2,283,228
Non-trainable params: 0
```

After creating a vocab after pre-processing the textual data, we need 2D tensors for both input and output. So, we to get all description sentences in equal shape, we need to do padding, that is if the sentence is longer than the set threshold (**max length**, hyperparameter set by us), we will clip it off assuming that all the important data must be lying at the beginning of the description. If the sentence is shorter than the threshold, we pad it with zeros. (**'0' padding**)

We will pass the sentences in batch-wise format.

So, the Input 2D Tensor will have dimensions: (BATCH_SIZE * MAXLEN OF BATCH)

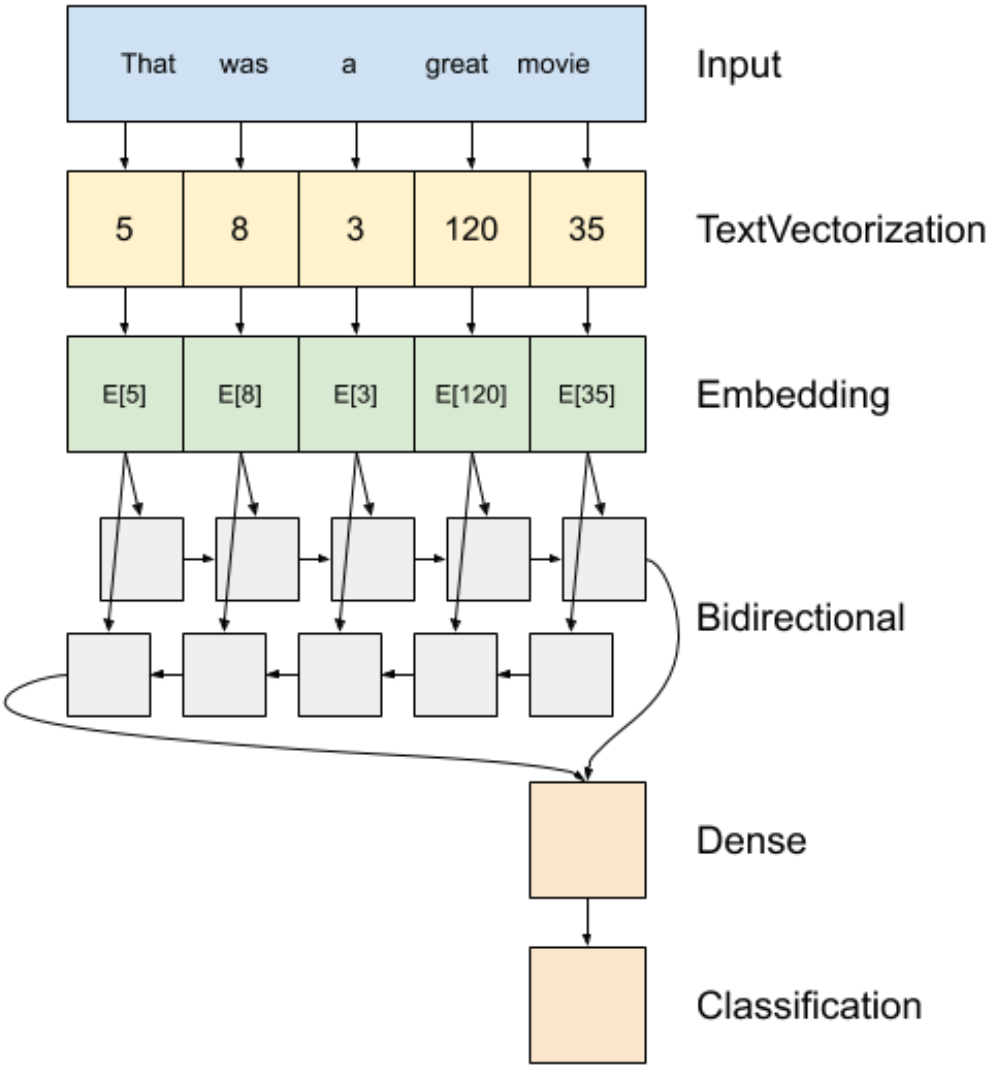
The Embedding Layer will be initialized with a random matrix of **|v| x k**

Where v is the fixed vocab size and corresponding to each index value in |v|, the 'k' sized vector embedding will be passed to RNN cell.

So, basically, for each word in vocab |v|, we have k-sized embedding that needs to be fed to LSTM layer for training.

Embedding layer will produce a 3D output of dimensions: ***(BATCH_SIZE * MAXLEN *k)***

Our **Model Architecture** would resemble this given image:



Source: https://www.tensorflow.org/tutorials/text/text_classification_rnn

Embedding Layer takes in parameters of Batch size, embedding size k and Max Length.

Our model has been given 22000, 100 and 300 respectively.

Next Layer is **Spatial Dropout 1D**:

It performs the same function as Dropout, however, it drops entire 1D feature maps instead of individual elements. If adjacent frames within feature maps are strongly correlated, then regular dropout will not regularize the activations and will otherwise just result in an effective learning rate decrease. In this case, SpatialDropout1D will help promote independence between feature maps and should be used instead. It takes in 3D tensor input from the previous Embedding Layer.

Next is the **LSTM layer** (Long Short-Term Memory layer): This layer will have 300 cells (because maxlen (number of words) = 300)

LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

We have taken 100 units (positive integer) It is basically the dimensionality of output space (Activation Output Vector)

And finally, we add a **Dense Layer**. Depending upon the number of output classes for classification, we add that many number of Dense Layers. In our case, we have 28 product categories. Therefore, 28 dense layers with activation as Softmax (multiclass classification). For a binary classification, we take Sigmoid activation.

5. Model Training

We have split our data into Training data (80%) and Testing data (20%).

The training has been done for Naïve Bayes with experiments on Vectorizers (Count vectorizer, TFIDF vectorizer) and n-grams for feature extractions.

The training for LSTM model has been done for 10 epochs initially with a batch size of 128 in each iteration. A validation split has also been maintained (20%) to prevent overfitting and monitoring our loss and accuracy.

For the LSTM model,

Accuracy is **93.6%** (initially) which seems good. We can improve this accuracy by training for more epochs and making sure that we do not overfit the model by training too much.

So, we will maintain two call backs (functionality provided by Keras):

1. **Model Checkpoint** -- to save the best model based on provided metric (helps prevent overfitting)
2. **Early Stopping** -- to stop training when a monitored quantity has stopped improving (helps in saving time)

6. Model Accuracy and Observations:

- **Naïve Bayes Classifier Results:**

I. *Multinomial NB with TFIDF Vectorizer:*

Training Accuracy: 86.81%
Test Accuracy: 84.43%

II. *Multinomial NB with Count Vectorizer with ngram=(1,2)*

Training Accuracy: 94.98%
Test Accuracy: 91.34%

III. *Multinomial NB with Count Vectorizer with ngram=(2,4):*

Training Accuracy: 94.46%
Test Accuracy: 89.19%

IV. *Multinomial NB with Count Vectorizer with default ngram=(1,1)*

Training Accuracy: 94.76%
Test Accuracy: 92.68%

Other Model Metrics:

These have been calculated for Naïve Bayes algorithm (in Notebook 2)

Precision

Precision is a good measure to determine, when the costs of False Positive is high. It's the percentage of examples that are Positive and are really classified as Positive. (sum along the column)

Recall

Recall calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive). (sum along the row)

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Goal of the model is to achieve high values of Precision and Recall.

Confusion Matrix

This matrix represents the relationship between Actual and Predicted labels (whether they are truly predicted as what they were labeled or not)

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

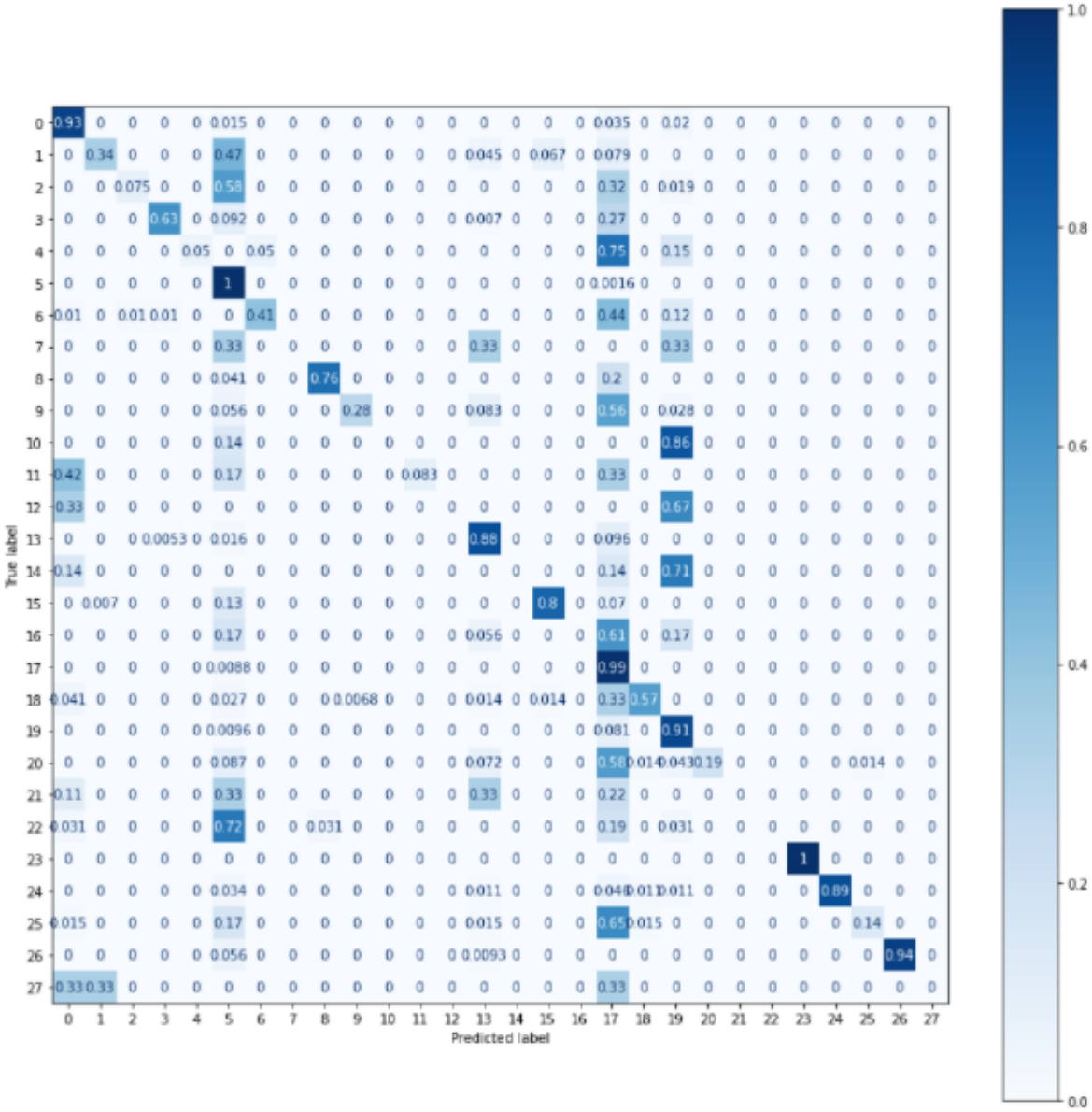
Conclusion:

Clearly the default ngram range of 1,1 worked the best with Count Vectorizer. This suggests that every word as a feature was important and there was no need of TFIDF here because every word's occurrence was important and contributed to uniqueness of each category.

Hence the best Naïve Bayes classifier has an accuracy of **92.68%** on Test Data.

Confusion Matrix for Naïve Bayes obtained:

We can clearly see some False Positives and False Negatives which suggest some mispredictions. But we also have 90%+ cases along the diagonal which show True Positives and True Negatives.



- **LSTM Model Results:**

1. *First Training with 10 Epochs and no Callbacks:*

   Training Accuracy: 97.72%
   Test Accuracy: 93.60%

2. *Second Training on saved Model with two Callbacks:*

   Training Accuracy: 98.60%
   Test Accuracy: 94.90%

Observation:

1. Early Stopping callback stopped the training after Epoch 5 of 10 because our chosen metric **"Validation Accuracy"** stopped improving and started dropping (which could have led to Overfitting and increased training accuracy, thereby performing bad on Test Data).

2. Model Checkpoint is saving the best statistics model with the best possible training and validation accuracy by primarily considering the minimum **"Validation Loss"** in a given Epoch.

Conclusion:

The best Accuracy on our **Test Data** is **94.9%**

It has improved in comparison to the previous accuracy of 93.6%. This is a significant improvement, and the model performs well enough to categorize for 28 labels.

Future Scope:

After going through the descriptions and the predicted labels by our model (Predictions shown in the Notebook), we can conclude that our model is performing quite accurately.

We can try to improve the model by experimenting with hyperparameters and changing or adding few more layers to the LSTM model.

We might see a bit more accuracy, but we must make sure that it doesn't overfit. Otherwise, our model will perform poorly on unseen and new dataset related to product categories.

7. Predictions:

Some of the example predictions from Notebook:

```
['feature leading lady woman camisole fabric cotton brand color black beig
e type camisole sale package leading lady woman camisole price spice colle
ction funky spaghetti camisole versatile enough worn innerwear loungewear
ribbed superior stretch body perfectly slip short unbuttoned casual look s
pecification leading lady woman camisole camisole slip detail number conte
nt sale package pack fabric cotton type camisole neck racerback neck gener
al detail pattern solid ideal woman fabric care gentle machine wash lukewa
rm water bleach']

Predicted Label:
 Clothing

['feature baby girl trouser occasion casual specification baby girl trouse
r general detail occasion casual ideal girl additional detail style code y
bfw311']

Predicted Label:
 Baby Care

['specification anand archies girl flat general ideal girl occasion casual
sandal detail sole material type flat heel height inch outer material arti
ficial leather color pink sandal']

Predicted Label:
 Footwear
```

8. References:

1. https://towardsdatascience.com/multi-class-text-classification-with-deep-learning-using-bert-b59ca2f5c613
2. https://www.kaggle.com/PromptCloudHQ/flipkart-products
3. https://towardsdatascience.com/ml-powered-product-categorization-for-smart-shopping-options-8f10d78e3294
4. http://publish.illinois.edu/mvauhkonen/files/2016/04/GottipatiVauhkonen-EcommerceProductCategorization.pdf
5. https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c
6. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

I certify that I have went through these articles and links in order to bring my original work to practicality.

-Pankhuri Bhatnagar