TASK 3: NLP

Name: Shaney Waris | **Roll no**.: 2018308 | **Email**: shaney18308@iiitd.ac.in | **WebApp**: Click here

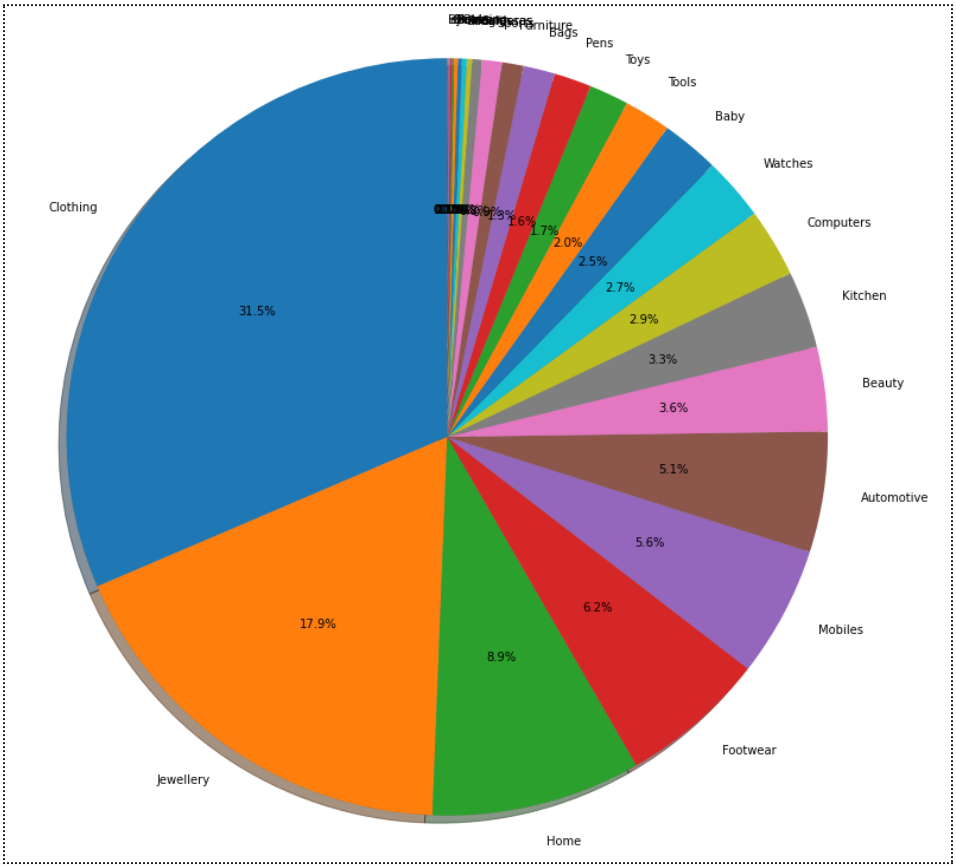
Step Wise Explanation:

1. Firstly I had imported all the necessary libraries.
2. Read the dataset using pandas and then extract the “primary product category” from the **product_category_tree** . So suppose below is the product category tree:

   "*Clothing >> Women's Clothing >> Lingerie*”

   Then I had taken “***Clothing***” as the primary product category.

3. Counted all the distinct primary categories and their occurrences in the whole dataset. So below is the pie chart of all the distinct categories that exist in the dataset with their respective percentage of occurrence:



In total, I got 202 different primary categories in the whole dataset, and out of these 202 categories, only 26 categories have occurrences greater than and equals to 10.
Other 202 - 26 = 176 categories have occurrences lesser than 10 in the whole dataset. So I have ignored these 176 categories because I don't have enough data for these categories and it will eventually decrease the performance of my ML model.

So my final 26 categories with their respective occurrences are:

```
Clothing      6198
Jewellery     3531
Home          1753
Footwear      1227
Mobiles       1099
Automotive    1012
Beauty         710
Kitchen        647
Computers      578
Watches        530
Baby           483
Tools          391
Toys           330
Pens           313
Bags           265
Furniture      180
Sports         166
Cameras         82
Health          43
Sunglasses      40
Gaming          35
Pet             30
Clovia          17
Olvin           16
eBooks          15
Eyewear         10
```

Sum of the occurrences of all these 26 categories = 19701.

So, initially, my dataframe shape was (20000, 18) and after considering only these 26 categories, my dataframe shape became (19701, 18).
Hence this proves that the remaining 176 categories are occurring in only 20000 - 19701 = 299 rows, which is very lesser data for 176 distinct categories.

4. Feature Vector creation.

I have tried out the below two different feature vectors and measured the accuracy in both cases:

- When only product description column considered in the feature vector.
- When “Product Description” + “Product name” + “Product brand name” + “Product retail price” considered in the feature vector.
    - **Reason**: Product name, Product brand name, Product retail price also plays a major role to decide on which category that product belongs to. So for example, clothes belong to some fixed set of brands.

5. Handling the NaN values in the dataframe/dataset:

Initially,
Retail price column had 75 NaN values,
Description had 2 NaN Values,
Brand had 5864 NaN values,
Product name had 0 NaN values.

Now, I have replaced the NaN value in the:
Retail price column to 0 values,
Description to empty string (“”)
Brand to empty string.

6. Preprocessing the Data:

This is one of the most important step in my whole process, the more efficiently and finely the preprocessing will be done, the more accuracy our model will give.

So, below are the steps for the data processing I did:

a. **Lowercasing all the content**.
   Reason: So that my models treat the same words but with different cases, for example: to treat “the” and “The” word the same.

b. **Converting numbers into words and removing the words having length equals to 1**.
   Reason: As I am also considering the retails price, because it will give me the valuable information (or a price range) that product belongs to. Hence I have also taken retails price into account and converted it into words.
   Words with length one don’t give me much valuable information (I had tried it out and the accuracy decreases when considering these single length words, it gives me

accuracy around 83%); hence I had removed them. It gradually increases the accuracy of my model up to 97%.

c. **Word tokenizes the content and removes the stopwords present in it.**
   Reason: These stopwords don't give me any valuable information, hence I removed them.

d. **Removed the URLs and punctuations present in the content**:
   Reason: Again, these URLs and punctuations don't give me any valuable information, hence I removed them.

e. **Lemmatize the content:**
   Reason: To convert the words into their root words, as their meaning is almost the same.

f. **Normalize the content**:
   Reason: So suppose there are some elongated words present in the content, then I just convert them into their root words. For example: converted “hellooooooo” to “hello”, etc.

g. And lastly, again **repeat step b**, because after the preprocessing it might be possible that some number becomes a word and there exist some single length words.

7. Different ML models Used:

Since this is more like a classification problem, so I have used below seven different types of classifiers in existing in ML/DL:

I. Random Forest Classifier.
II. Decision Tree.
III. Perceptron
IV. MLP Classifier.
V. Logistic Regression.
VI. LinearSVC.
VII. K nearest neighbours.

8. Different Approaches to create the Numerical Feature Vectors from Text:

I had discovered many approaches to create vectors and analysed which approach will give me better results. So in total, there are 10 different approaches I had used to create the vectors. All the approaches are there in my Jupiter notebook.

I. TF-IDF Vectorization.
II. Bag of Words Vectorization
III. Binary Vectorization.
IV. Google’s Universal Sentence Encoder.
   Ref: https://tfhub.dev/google/universal-sentence-encoder/1

V. Sentence Transformer.
Ref: https://huggingface.co/sentence-transformers

VI. Bert pretrained model.
Ref: https://huggingface.co/transformers/model_doc/bert.html

VII. SqueezeBert pretrained model.
Ref: https://huggingface.co/transformers/model_doc/squeezebert.html

VIII. DeBERTa pretrained model.
Ref: https://huggingface.co/transformers/model_doc/deberta.html

IX. GPT2 pretrained model.
Ref: https://huggingface.co/transformers/model_doc/gpt2.html

X. T5 pretrained model.
Ref: https://huggingface.co/transformers/model_doc/t5.html

9. Model Evaluation.

To measure the accuracy of the model, I have split the whole data into 60:40 ratio for training and testing respectively. Trained my model on this 60% training data and testing on this 40% testing data. And hence reporting the accuracies.

Evaluation Metrics I have used:

- Accuracy.
- Precision
- Recall.
- F1 Score.

Reporting Accuracies for different approaches and models:

A. When ONLY "Description" is taken in a Feature Vector:

I. For TF-IDF Vectorization Technique:

Random Forest: Accuracy = 0.9427848101265823 Precision = 0.9430948602902899 Recall = 0.9427848101265823 F1 score = 0.9400577403440608	**Decision Tree**: Accuracy = 0.929367088607595 Precision = 0.9296308838165671 Recall = 0.929367088607595 F1 score = 0.9284085085166722
Perceptron: Accuracy = 0.9616455696202532 Precision = 0.9595856694472494 Recall = 0.9616455696202532 F1 score = 0.959844476416066	**MLPClassfier**: Accuracy = 0.970126582278481 Precision = 0.9702556570270183 Recall = 0.970126582278481 F1 score = 0.9694045657467953

Logistic Regression: Accuracy = 0.9464556962025317 Precision = 0.9445351198177433 Recall = 0.9464556962025317 F1 score = 0.9419825626613967	**Linear SVC**: Accuracy = 0.9713924050632912 Precision = 0.9708105616134443 Recall = 0.9713924050632912 F1 score = 0.9704041429963418
K Nearest Neighbours: Accuracy = 0.9473417721518987 Precision = 0.947310194500963 Recall = 0.9473417721518987 F1 score = 0.9460605873107389	

II. For Bag of Words Vectorization Technique:

Random Forest: Accuracy = 0.9486075949367089 Precision = 0.9490304349277892 Recall = 0.9486075949367089 F1 score = 0.9460440799208318	**Decision Tree**: Accuracy = 0.9382278481012658 Precision = 0.9387230896746445 Recall = 0.9382278481012658 F1 score = 0.9377415263079389
Perceptron: Accuracy = 0.9632911392405064 Precision = 0.9658289693531817 Recall = 0.9632911392405064 F1 score = 0.9635725280560121	**MLPClassfier**: Accuracy = 0.9667088607594937 Precision = 0.9675122471791007 Recall = 0.9667088607594937 F1 score = 0.9664313627635043
Logistic Regression: Accuracy = 0.9648101265822785 Precision = 0.9641724033298784 Recall = 0.9648101265822785 F1 score = 0.9639100875738963	**Linear SVC**: Accuracy = 0.9662025316455696 Precision = 0.9672939967275597 Recall = 0.9662025316455696 F1 score = 0.9662720452738921
K Nearest Neighbours: Accuracy = 0.9078481012658228 Precision = 0.9082258678037753 Recall = 0.9078481012658228 F1 score = 0.9064413400469529	

III. For Binary Vectorization Technique:

Random Forest: Accuracy = 0.9502531645569621 Precision = 0.9503681086930525 Recall = 0.9502531645569621 F1 score = 0.9478847862029193	**Decision Tree**: Accuracy = 0.9408860759493671 Precision = 0.9410361464758648 Recall = 0.9408860759493671 F1 score = 0.9402450447023599
Perceptron: Accuracy = 0.9660759493670886 Precision = 0.9675599398623186 Recall = 0.9660759493670886 F1 score = 0.9656999357951295	**MLPClassfier**: Accuracy = 0.9693670886075949 Precision = 0.968963910757877 Recall = 0.9693670886075949 F1 score = 0.9684013718033347
Logistic Regression: Accuracy = 0.9669620253164557 Precision = 0.9654033605010993 Recall = 0.9669620253164557 F1 score = 0.9656857260641962	**Linear SVC**: Accuracy = 0.970632911392405 Precision = 0.9699530095608161 Recall = 0.970632911392405 F1 score = 0.9699201970706507
K Nearest Neighbours: Accuracy = 0.910506329113924 Precision = 0.9103463048994717 Recall = 0.910506329113924 F1 score = 0.9078081114765638	

IV. For Google's Universal Sentence Encoder:

Random Forest: Accuracy = 0.9335443037974683 Precision = 0.9356514563445846 Recall = 0.9335443037974683 F1 score = 0.9288259572368369	**Decision Tree**: Accuracy = 0.8384810126582278 Precision = 0.8397178118369889 Recall = 0.8384810126582278 F1 score = 0.8380796873009219
Perceptron: Accuracy = 0.9477215189873418 Precision = 0.9559469339254901 Recall = 0.9477215189873418 F1 score = 0.9509163292039551	**MLPClassfier**: Accuracy = 0.9640506329113924 Precision = 0.9651831947507782 Recall = 0.9640506329113924 F1 score = 0.9640771165198452

Logistic Regression: Accuracy = 0.9430379746835443 Precision = 0.9388285624727138 Recall = 0.9430379746835443 F1 score = 0.9388464835962491	**Linear SVC**: Accuracy = 0.959873417721519 Precision = 0.9578012364730337 Recall = 0.959873417721519 F1 score = 0.9581758069670605
K Nearest Neighbours: Accuracy = 0.9563291139240506 Precision = 0.9558158993277593 Recall = 0.9563291139240506 F1 score = 0.9553227509585818	

V. For Sentence Transformer:

Random Forest: Accuracy = 0.8968354430379747 Precision = 0.9050713700883612 Recall = 0.8968354430379747 F1 score = 0.8893741716192891	**Decision Tree**: Accuracy = 0.7536708860759493 Precision = 0.7559204447533808 Recall = 0.7536708860759493 F1 score = 0.7541774738844731
Perceptron: Accuracy = 0.9418987341772151 Precision = 0.9544220300565461 Recall = 0.9418987341772151 F1 score = 0.9459879620594256	**MLPClassfier**: Accuracy = 0.9554430379746836 Precision = 0.9555889741150785 Recall = 0.9554430379746836 F1 score = 0.9547924633018172
Logistic Regression: Accuracy = 0.9548101265822785 Precision = 0.9533416483093375 Recall = 0.9548101265822785 F1 score = 0.953646708648532	**Linear SVC**: Accuracy = 0.9536708860759494 Precision = 0.9544744585030308 Recall = 0.9536708860759494 F1 score = 0.9533464006455425
K Nearest Neighbours: Accuracy = 0.9431645569620253 Precision = 0.9425845510979904 Recall = 0.9431645569620253 F1 score = 0.941670748998123	

VI. For Bert Transformer:

Random Forest: Accuracy = 0.7364556962025316 Precision = 0.7649124368089227 Recall = 0.7364556962025316 F1 score = 0.7213482518969369	**Decision Tree**: Accuracy = 0.6708860759493671 Precision = 0.6695960410694926 Recall = 0.6708860759493671 F1 score = 0.6695914757376681
Perceptron: Accuracy = 0.32367088607594935 Precision = 0.36829540150251844 Recall = 0.32367088607594935 F1 score = 0.31702888442349486	**MLPClassfier**: Accuracy = 0.32050632911392407 Precision = 0.33421127262195793 Recall = 0.32050632911392407 F1 score = 0.17165694970319767
Logistic Regression: Accuracy = 0.45924050632911395 Precision = 0.4240488161821474 Recall = 0.45924050632911395 F1 score = 0.41797060719652246	**Linear SVC**: Accuracy = 0.20556962025316455 Precision = 0.3572232250203189 Recall = 0.20556962025316455 F1 score = 0.19763360232901153
K Nearest Neighbours: Accuracy = 0.6306329113924051 Precision = 0.6456114815902327 Recall = 0.6306329113924051 F1 score = 0.6195879347496565	

VII. For SqueezeBert Transformer:

Random Forest: Accuracy = 0.7483544303797468 Precision = 0.7790219329034394 Recall = 0.7483544303797468 F1 score = 0.7349150461824391	**Decision Tree**: Accuracy = 0.6732911392405063 Precision = 0.6738877960126037 Recall = 0.6732911392405063 F1 score = 0.6730970563194117
Perceptron: Accuracy = 0.3382278481012658 Precision = 0.38950625177251486 Recall = 0.3382278481012658 F1 score = 0.3205469767578448	**MLPClassfier**: Accuracy = 0.30658227848101266 Precision = 0.2641001189017937 Recall = 0.30658227848101266 F1 score = 0.148507989605804

Logistic Regression: Accuracy = 0.45354430379746835 Precision = 0.417669351564032 Recall = 0.45354430379746835 F1 score = 0.4108659091542787	**Linear SVC**: Accuracy = 0.33582278481012656 Precision = 0.3683129578683723 Recall = 0.33582278481012656 F1 score = 0.32599421350312835
K Nearest Neighbours: Accuracy = 0.6289873417721519 Precision = 0.6294808115056301 Recall = 0.6289873417721519 F1 score = 0.6133587487101323	

VIII. For DeBERTa Transformer:

Random Forest: Accuracy = 0.7491139240506329 Precision = 0.780758007335195 Recall = 0.7491139240506329 F1 score = 0.7354462595648906	**Decision Tree**: Accuracy = 0.6655696202531646 Precision = 0.6648168742832583 Recall = 0.6655696202531646 F1 score = 0.6642056813906372
Perceptron: Accuracy = 0.22974683544303798 Precision = 0.3644089678731215 Recall = 0.22974683544303798 F1 score = 0.22149629283342412	**MLPClassfier**: Accuracy = 0.31506329113924053 Precision = 0.3473044806611194 Recall = 0.31506329113924053 F1 score = 0.16399039821056577
Logistic Regression: Accuracy = 0.43886075949367087 Precision = 0.39876912426908245 Recall = 0.43886075949367087 F1 score = 0.3978436924882686	**Linear SVC**: Accuracy = 0.3015189873417721 Precision = 0.31243266789430274 Recall = 0.3015189873417721 F1 score = 0.2870539928207542
K Nearest Neighbours: Accuracy = 0.6127848101265823 Precision = 0.6218070428988267 Recall = 0.6127848101265823 F1 score = 0.5942120414802633	

IX. For GPT2 Transformer:

Random Forest: Accuracy = 0.7478481012658228 Precision = 0.7783132791830345 Recall = 0.7478481012658228 F1 score = 0.73328900366638	**Decision Tree**: Accuracy = 0.6741772151898734 Precision = 0.6700114407586758 Recall = 0.6741772151898734 F1 score = 0.6715313707815546
Perceptron: Accuracy = 0.25974683544303795 Precision = 0.30505159167564166 Recall = 0.25974683544303795 F1 score = 0.2446820035573327	**MLPClassfier**: Accuracy = 0.309746835443038 Precision = 0.27140304213997685 Recall = 0.309746835443038 F1 score = 0.15841753196698796
Logistic Regression: Accuracy = 0.44531645569620254 Precision = 0.39790208621030326 Recall = 0.44531645569620254 F1 score = 0.3967248512292167	**Linear SVC**: Accuracy = 0.17455696202531645 Precision = 0.33610582296314634 Recall = 0.17455696202531645 F1 score = 0.20356404687876956
K Nearest Neighbours: Accuracy = 0.6078481012658228 Precision = 0.608369297161118 Recall = 0.6078481012658228 F1 score = 0.5945112301978926	

X. For T5 Transformer:

Random Forest: Accuracy = 0.7189873417721518 Precision = 0.7553181531596362 Recall = 0.7189873417721518 F1 score = 0.7016339724553727	**Decision Tree**: Accuracy = 0.6482278481012658 Precision = 0.6452349562358878 Recall = 0.6482278481012658 F1 score = 0.6460611643823678
Perceptron: Accuracy = 0.3210126582278481 Precision = 0.3344204020587124 Recall = 0.3210126582278481 F1 score = 0.2898737138584534	**MLPClassfier**: Accuracy = 0.320253164556962 Precision = 0.3634321755088222 Recall = 0.320253164556962 F1 score = 0.17345224931008046

Logistic Regression: Accuracy = 0.4220253164556962 Precision = 0.38095339025460184 Recall = 0.4220253164556962 F1 score = 0.38040723650170366	**Linear SVC**: Accuracy = 0.2768354430379747 Precision = 0.2790951492037472 Recall = 0.2768354430379747 F1 score = 0.25931498941288933
K Nearest Neighbours: Accuracy = 0.5913924050632912 Precision = 0.584414545084847 Recall = 0.5913924050632912 F1 score = 0.5718920222454587	

B. When "Description", "Product Name", "Product Brand" and "Product Retail Price" are taken in a Feature Vector:

I. For TF-IDF Vectorization Technique:

Random Forest: Accuracy = 0.9442964090851415 Precision = 0.9441150539080287 Recall = 0.9442964090851415 F1 score = 0.941193263325551	**Decision Tree**: Accuracy = 0.9272934906737723 Precision = 0.9267922313192267 Recall = 0.9272934906737723 F1 score = 0.926424200397433
Perceptron: Accuracy = 0.9717040984646619 Precision = 0.9723024246585851 Recall = 0.9717040984646619 F1 score = 0.971578420900613	**MLPClassfier**: Accuracy = 0.973099860423804 Precision = 0.9729158167751014 Recall = 0.973099860423804 F1 score = 0.9724553335482413
Logistic Regression: Accuracy = 0.9524172059383327 Precision = 0.9506200854792585 Recall = 0.9524172059383327 F1 score = 0.9488263800771809	**Linear SVC**: Accuracy = 0.972972972972973 Precision = 0.9722766913538154 Recall = 0.972972972972973 F1 score = 0.9721483576404576
K Nearest Neighbours: Accuracy = 0.9474685953559193 Precision = 0.9454796624316452 Recall = 0.9474685953559193 F1 score = 0.946011579393558	

II. For Bag of Words Vectorization Technique:

Random Forest: Accuracy = 0.951655881233346 Precision = 0.9512536116639044 Recall = 0.951655881233346 F1 score = 0.9491266164334756	**Decision Tree**: Accuracy = 0.941504885166857 Precision = 0.9415828037422891 Recall = 0.941504885166857 F1 score = 0.9410275654070193
Perceptron: Accuracy = 0.9692932368988707 Precision = 0.9698153896443943 Recall = 0.9692932368988707 F1 score = 0.9692839820936227	**MLPClassfier**: Accuracy = 0.9705621114071818 Precision = 0.9707406020484023 Recall = 0.9705621114071818 F1 score = 0.9699714359144644
Logistic Regression: Accuracy = 0.9692932368988707 Precision = 0.9683886838982094 Recall = 0.9692932368988707 F1 score = 0.9686210685165437	**Linear SVC**: Accuracy = 0.9698007867021952 Precision = 0.9704471352993924 Recall = 0.9698007867021952 F1 score = 0.9695790893863445
K Nearest Neighbours: Accuracy = 0.91561984519731 Precision = 0.9218672429105828 Recall = 0.91561984519731 F1 score = 0.915049384793465	

III. For Binary Vectorization Technique:

Random Forest: Accuracy = 0.9491181322167238 Precision = 0.9485607525339077 Recall = 0.9491181322167238 F1 score = 0.946612358431322	**Decision Tree**: Accuracy = 0.943788859281817 Precision = 0.9430860298284386 Recall = 0.943788859281817 F1 score = 0.9429572764232206
Perceptron: Accuracy = 0.9695470118005329 Precision = 0.9696711002722737 Recall = 0.9695470118005329 F1 score = 0.9693990941900301	**MLPClassfier**: Accuracy = 0.9717040984646619 Precision = 0.970675937983725 Recall = 0.9717040984646619 F1 score = 0.9708889412810922

Logistic Regression: Accuracy = 0.967389925136404 Precision = 0.9660572493591276 Recall = 0.967389925136404 F1 score = 0.9661877754572051	**Linear SVC**: Accuracy = 0.9720847608171552 Precision = 0.9719922075005393 Recall = 0.9720847608171552 F1 score = 0.9714969776382797
K Nearest Neighbours: Accuracy = 0.9162542824514656 Precision = 0.9174190143794947 Recall = 0.9162542824514656 F1 score = 0.9135851528254034	

IV. For Google's Universal Sentence Encoder:

Random Forest: Accuracy = 0.9379520365435858 Precision = 0.9391854616371188 Recall = 0.9379520365435858 F1 score = 0.9337668439147148	**Decision Tree**: Accuracy = 0.8368227382311889 Precision = 0.8365399303866579 Recall = 0.8368227382311889 F1 score = 0.8356942196431996
Perceptron: Accuracy = 0.9521634310366704 Precision = 0.9555604952972976 Recall = 0.9521634310366704 F1 score = 0.9519372720554338	**MLPClassfier**: Accuracy = 0.9626950894556529 Precision = 0.9625313563850786 Recall = 0.9626950894556529 F1 score = 0.9622903024927383
Logistic Regression: Accuracy = 0.9456921710442837 Precision = 0.9412892205542329 Recall = 0.9456921710442837 F1 score = 0.9423166525974378	**Linear SVC**: Accuracy = 0.9620606522014973 Precision = 0.9606008030538958 Recall = 0.9620606522014973 F1 score = 0.9607067366784514
K Nearest Neighbours: Accuracy = 0.9600304529881994 Precision = 0.9593213492692065 Recall = 0.9600304529881994 F1 score = 0.9592825989443617	

V. For Sentence Transformer:

Random Forest: Accuracy = 0.9006471259992387 Precision = 0.905378367789267 Recall = 0.9006471259992387 F1 score = 0.8933013848054279	**Decision Tree**: Accuracy = 0.7472401979444233 Precision = 0.7522893153934509 Recall = 0.7472401979444233 F1 score = 0.7491381176245308
Perceptron: Accuracy = 0.9468341581017637 Precision = 0.9484392900929008 Recall = 0.9468341581017637 F1 score = 0.946669652715196	**MLPClassfier**: Accuracy = 0.9544474051516305 Precision = 0.954136529699897 Recall = 0.9544474051516305 F1 score = 0.9535105036146264
Logistic Regression: Accuracy = 0.9512752188808526 Precision = 0.9500974029349805 Recall = 0.9512752188808526 F1 score = 0.9502708522622285	**Linear SVC**: Accuracy = 0.9530516431924883 Precision = 0.9528247057179836 Recall = 0.9530516431924883 F1 score = 0.9526391454040974
K Nearest Neighbours: Accuracy = 0.9442964090851415 Precision = 0.944089278497224 Recall = 0.9442964090851415 F1 score = 0.9432048626186117	

VI. For Bert Transformer:

Random Forest: Accuracy = 0.7329019160005076 Precision = 0.7742606210797025 Recall = 0.7329019160005076 F1 score = 0.7179308896167004	**Decision Tree**: Accuracy = 0.650171298058622 Precision = 0.6514881536584022 Recall = 0.650171298058622 F1 score = 0.6496873287614647
Perceptron: Accuracy = 0.3254663113818043 Precision = 0.3494821685272512 Recall = 0.3254663113818043 F1 score = 0.2668875764712022	**MLPClassfier**: Accuracy = 0.3277502854967644 Precision = 0.36629593095704727 Recall = 0.3277502854967644 F1 score = 0.17809044533387358

Logistic Regression: Accuracy = 0.4531150869179038 Precision = 0.4102259321847847 Recall = 0.4531150869179038 F1 score = 0.4066426820083972	**Linear SVC**: Accuracy = 0.2820708031975638 Precision = 0.29768320983802726 Recall = 0.2820708031975638 F1 score = 0.27296425459938267
K Nearest Neighbours: Accuracy = 0.6091866514401726 Precision = 0.6349570967159459 Recall = 0.6091866514401726 F1 score = 0.6017903965483677	

VII. For SqueezeBert Transformer:

Random Forest: Accuracy = 0.7434335744194899 Precision = 0.7748260009643779 Recall = 0.7434335744194899 F1 score = 0.7272606533209247	**Decision Tree**: Accuracy = 0.6613373937317599 Precision = 0.6618229467334474 Recall = 0.6613373937317599 F1 score = 0.6607487175175567
Perceptron: Accuracy = 0.2828321279025504 Precision = 0.3808105628741853 Recall = 0.2828321279025504 F1 score = 0.27188977369783046	**MLPClassfier**: Accuracy = 0.3207714757010532 Precision = 0.2686852870251132 Recall = 0.3207714757010532 F1 score = 0.16362909299112613
Logistic Regression: Accuracy = 0.4703717802309352 Precision = 0.44035515516817775 Recall = 0.4703717802309352 F1 score = 0.43566089382476353	**Linear SVC**: Accuracy = 0.3071945184621241 Precision = 0.34844229753954814 Recall = 0.3071945184621241 F1 score = 0.3123488176386777
K Nearest Neighbours: Accuracy = 0.614135262022586 Precision = 0.6294335409865455 Recall = 0.614135262022586 F1 score = 0.6021073203372228	

VIII. For DeBERTa Transformer:

Random Forest: Accuracy = 0.740134500697881 Precision = 0.7762862259012913 Recall = 0.740134500697881 F1 score = 0.7240639697173216	**Decision Tree**: Accuracy = 0.6532165968785687 Precision = 0.6548676525712784 Recall = 0.6532165968785687 F1 score = 0.6534045797664059
Perceptron: Accuracy = 0.27788351732013705 Precision = 0.3808074647308704 Recall = 0.27788351732013705 F1 score = 0.29176347878327824	**MLPClassfier**: Accuracy = 0.32838472275091996 Precision = 0.3198778807520686 Recall = 0.32838472275091996 F1 score = 0.18178486522648712
Logistic Regression: Accuracy = 0.4537495241720594 Precision = 0.41306251014453294 Recall = 0.4537495241720594 F1 score = 0.41003896473850077	**Linear SVC**: Accuracy = 0.281182591041746 Precision = 0.3023943093525374 Recall = 0.281182591041746 F1 score = 0.2863778163766661
K Nearest Neighbours: Accuracy = 0.6117244004567948 Precision = 0.6354880247314776 Recall = 0.6117244004567948 F1 score = 0.5925043026742672	

IX. For GPT2 Transformer:

Random Forest: Accuracy = 0.7454637736327877 Precision = 0.7764315798081494 Recall = 0.7454637736327877 F1 score = 0.7299237971597717	**Decision Tree**: Accuracy = 0.6752950133231823 Precision = 0.6756648070601241 Recall = 0.6752950133231823 F1 score = 0.6747358140212473
Perceptron: Accuracy = 0.3399314807765512 Precision = 0.37674165104128476 Recall = 0.3399314807765512 F1 score = 0.3121812949595168	**MLPClassfier**: Accuracy = 0.33460220784164446 Precision = 0.36254320195117473 Recall = 0.33460220784164446 F1 score = 0.19579320317193974

Logistic Regression: Accuracy = 0.46148965867275726 Precision = 0.4161359467259394 Recall = 0.46148965867275726 F1 score = 0.4165074958988606	**Linear SVC**: Accuracy = 0.33929704352239565 Precision = 0.3049660580275905 Recall = 0.33929704352239565 F1 score = 0.283556183543751
K Nearest Neighbours: Accuracy = 0.6069026773252125 Precision = 0.613838109942647 Recall = 0.6069026773252125 F1 score = 0.5944255829563194	

X. For T5 Transformer:

Random Forest: Accuracy = 0.7417840375586855 Precision = 0.772808072495556 Recall = 0.7417840375586855 F1 score = 0.7260472080293012	**Decision Tree**: Accuracy = 0.6773252125364801 Precision = 0.677112974786427 Recall = 0.6773252125364801 F1 score = 0.6764975651934179
Perceptron: Accuracy = 0.3399314807765512 Precision = 0.37674165104128476 Recall = 0.3399314807765512 F1 score = 0.3121812949595168	**MLPClassfier**: Accuracy = 0.323943661971831 Precision = 0.3587814364591565 Recall = 0.323943661971831 F1 score = 0.17668685619866542
Logistic Regression: Accuracy = 0.46148965867275726 Precision = 0.4161359467259394 Recall = 0.46148965867275726 F1 score = 0.4165074958988606	**Linear SVC**: Accuracy = 0.23055449816013196 Precision = 0.35409701639880986 Recall = 0.23055449816013196 F1 score = 0.24711585114868423
K Nearest Neighbours: Accuracy = 0.6069026773252125 Precision = 0.613838109942647 Recall = 0.6069026773252125 F1 score = 0.5944255829563194	

Ideas I have to improve the accuracy of the model further:

Initially, I have tried to use pre-trained models to create feature vectors, because these models also take context into account while creating the feature vectors. But the accuracy they gave is very bad. This may be because the data is not enough for these pre-trained models.
Then I have tried to create vectors using TF-IDF, Bag of Words and Binary vectorization techniques to improve the accuracy of the model. And after testing, these 3 techniques actually improved the accuracy. So the maximum accuracy I got is 97.3099860423804%.

So, the total numbers of techniques I have tried =
(2 types of feature vector) X (**10 different approaches to create feature vectors)** X **(7 different ML models**)
= 140 techniques

And out of these 140 techniques, the best performance is given by "**MLP Classifier**" with "**TF-IDF Vectorization Technique**" when the feature vector is created with Description, Product Name, Product Brand, Product Retail Price with an accuracy of about 97.3099860423804%.

One thing I observed after doing many experiments is:
The best accuracy I got when feature vector is created with "Description" only is:

- 97.13924050632912% from the LinearSVC classifier with TF-IDF Vectorization Technique

The best accuracy I got when feature vector is created with "Description", "Product Name", "Product Brand", and "Product Retail Price" is:

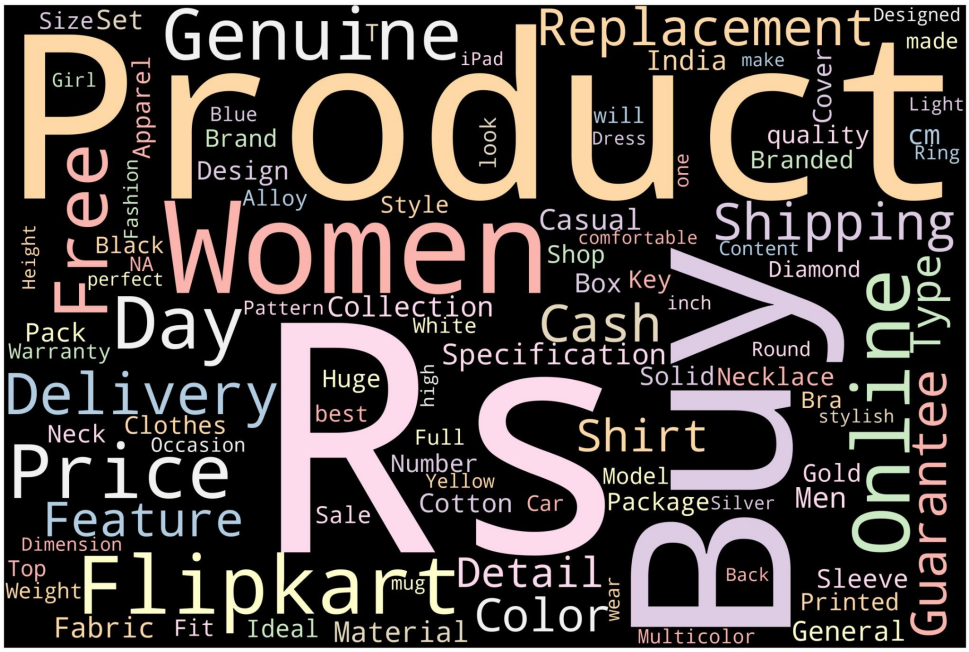
- 97.3099860423804% from the MLP classifier with TF-IDF Vectorization Technique

So the accuracy is not much increased when I have taken "Product Name", "Product Brand" and "Product Retail Price" also in my feature vector.

10. Data Visualization.

Note: The data which I am going to visualize belongs to the “description” column only.

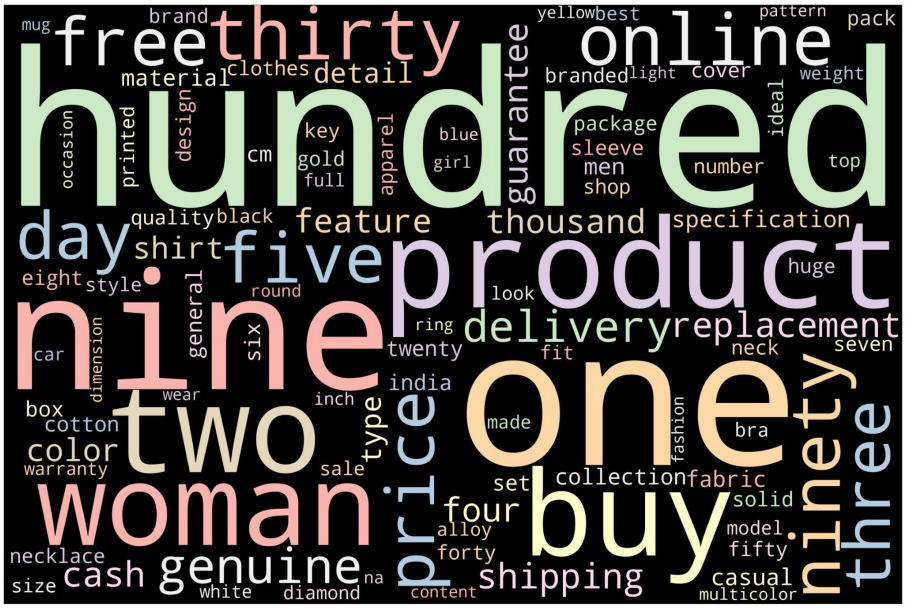
I. Word Cloud before preprocessing the data:



Insight:

- We can see some words like “NA”, “RS”, “cm”, “inch”, etc in this word cloud which is relevant enough to extract the information from them.

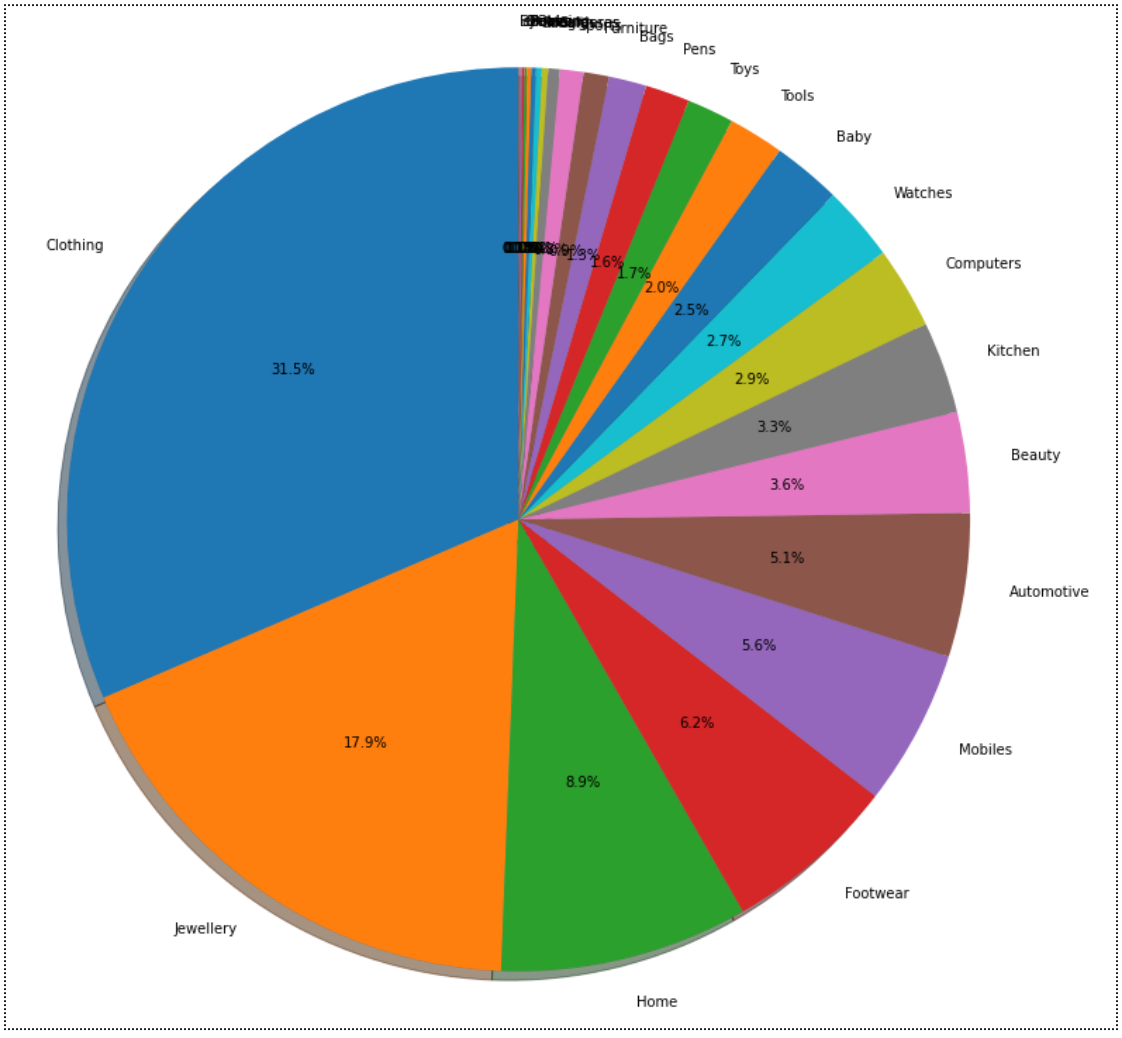
II. Word Cloud after preprocessing the data.



Insights:

- Now, after the preprocessing, no such irrelevant words are coming.
- We can see some words like:
    - “hundred”: Many items might have retail prices under some hundred rupees.
    - “women”: many items might be for women.
    - “free”: Maybe many items are free with some other items.

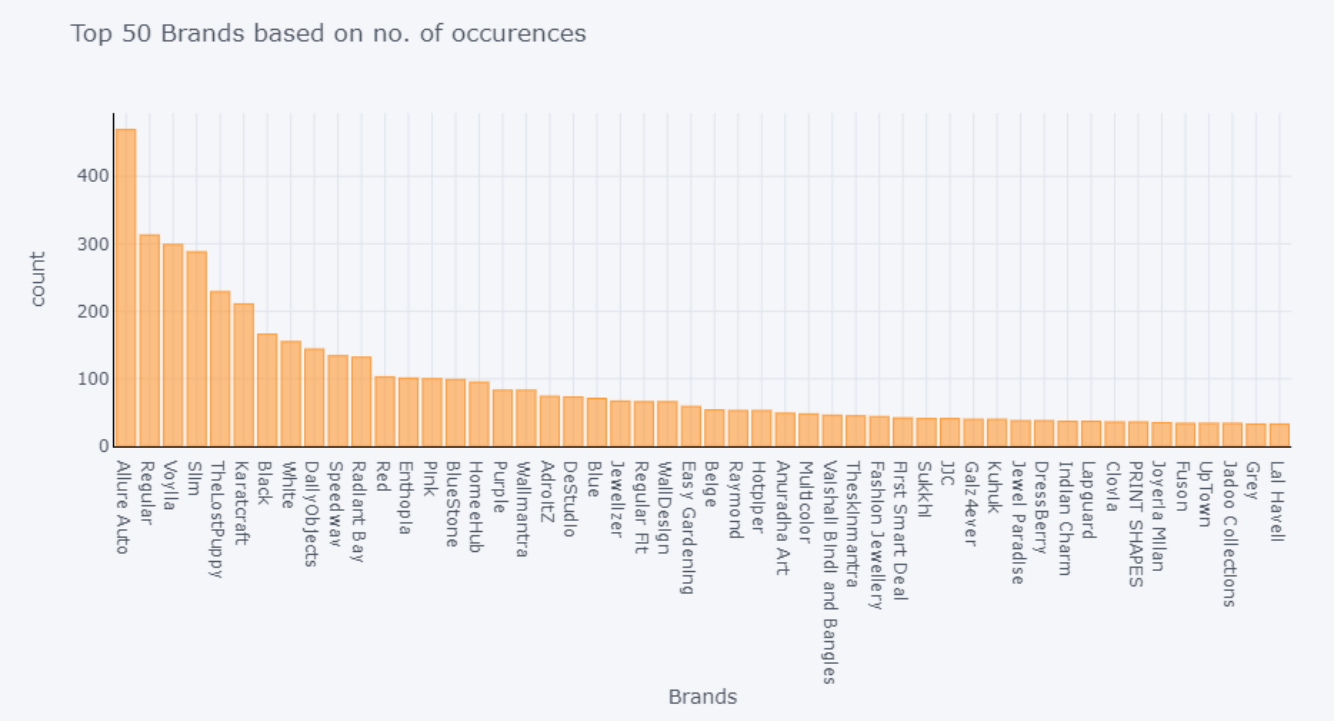
III. Product Categories Pie Chart:



Insights:

- Total distinct categories are 202.
- The 26 categories I have chosen (mentioned in step 2) covered almost 98.5% of dataset items.
- Other 1.5% dataset has remaining 176 distinct categories.

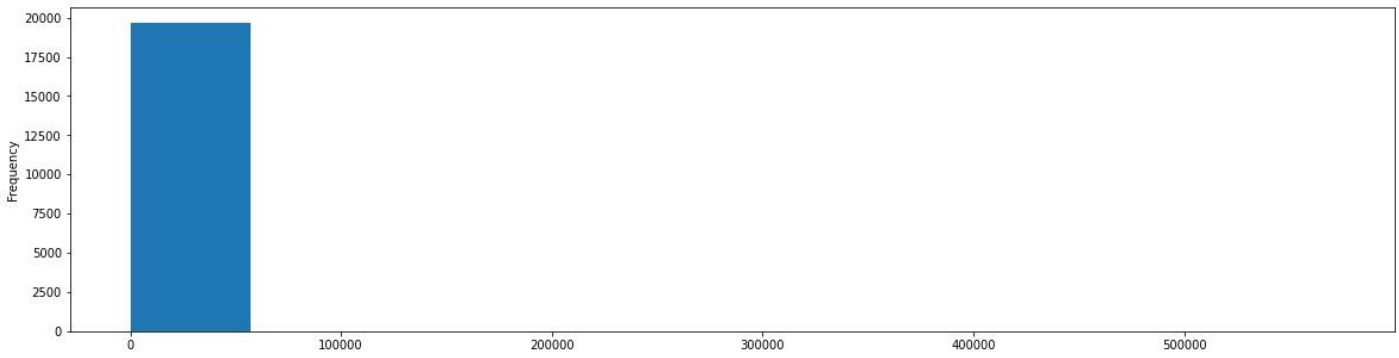
IV. Top 50 Brands based on no. of occurrences:



Insights:

- “Allure Auto” is the brand having the maximum number of distinct products (greater than 400) in the dataset. (This is just an image of an interactive plot, please check my notebook for a more detailed graph.)

V. Product retail price range distribution.

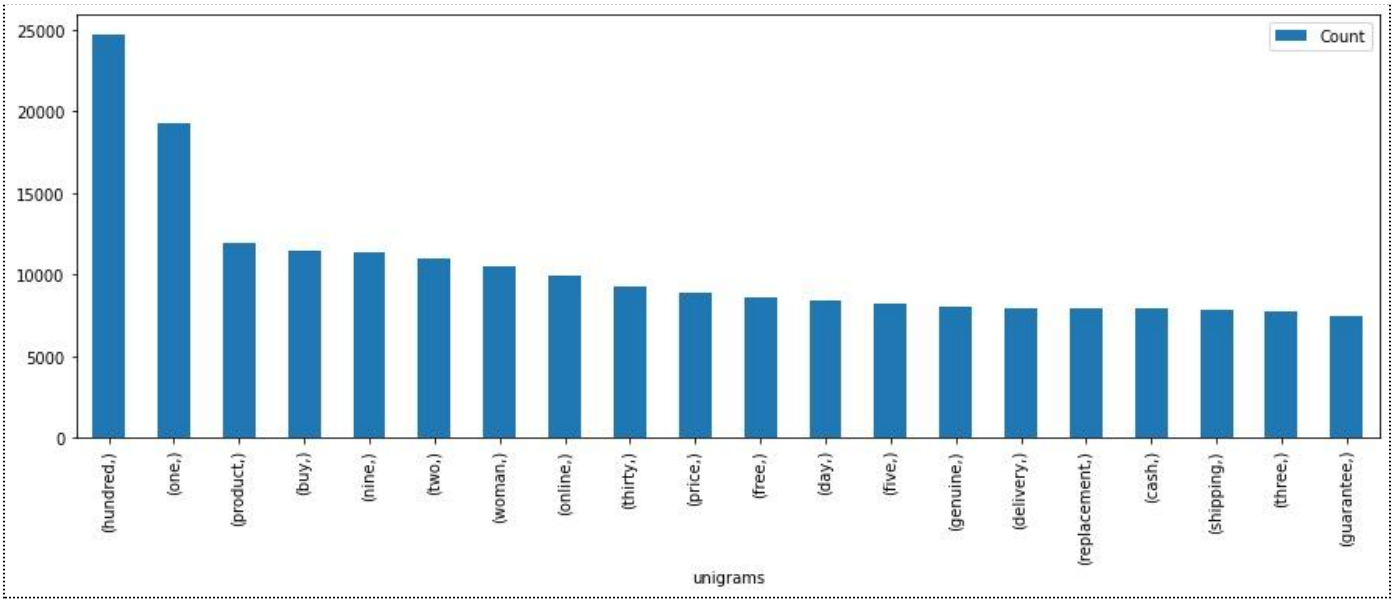


Insights:

- All the 20k products in the dataset have a retail price under 50k

Note: For a more detailed graph, please check the interactive plot for this in my “Data Visualization” notebook. It is not possible for me to put the interactive plot in the README.pdf file.

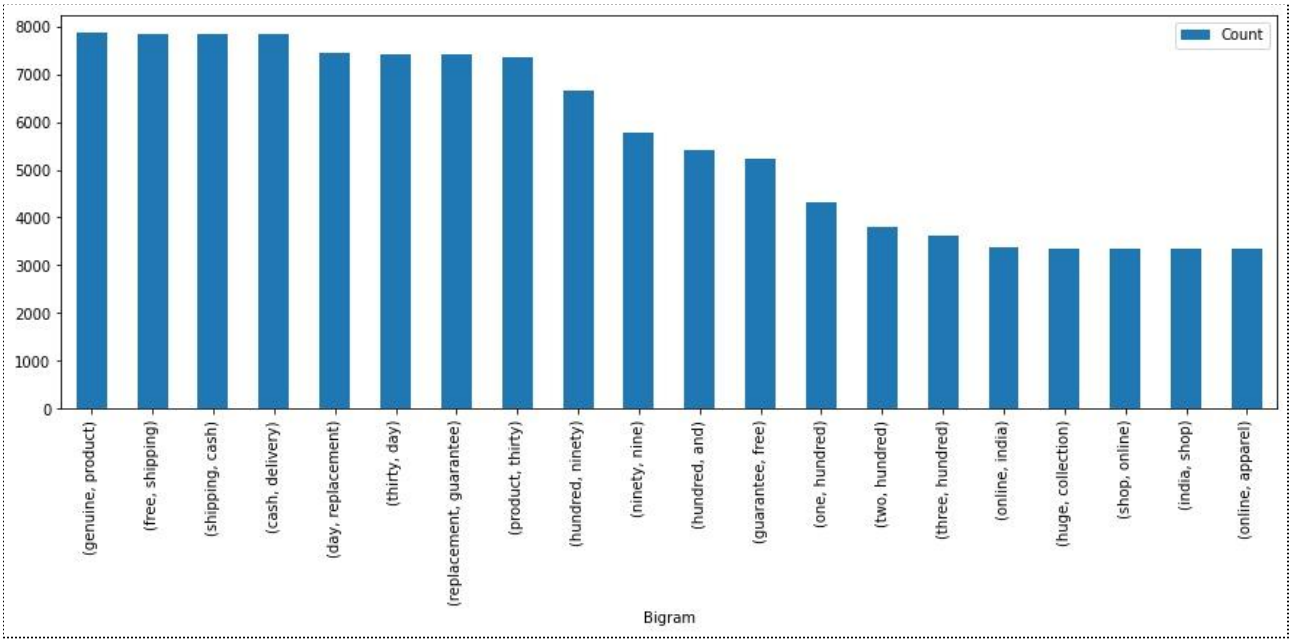
VI. Top 20 Unigrams distribution:



Insights:

- The most frequent word is “hundred” which means many items are under some hundred retail price rupees.
- It seems like many items are for “women”.
- Many items are providing the “replacement” as well.
- Many items must have some “guarantee”.

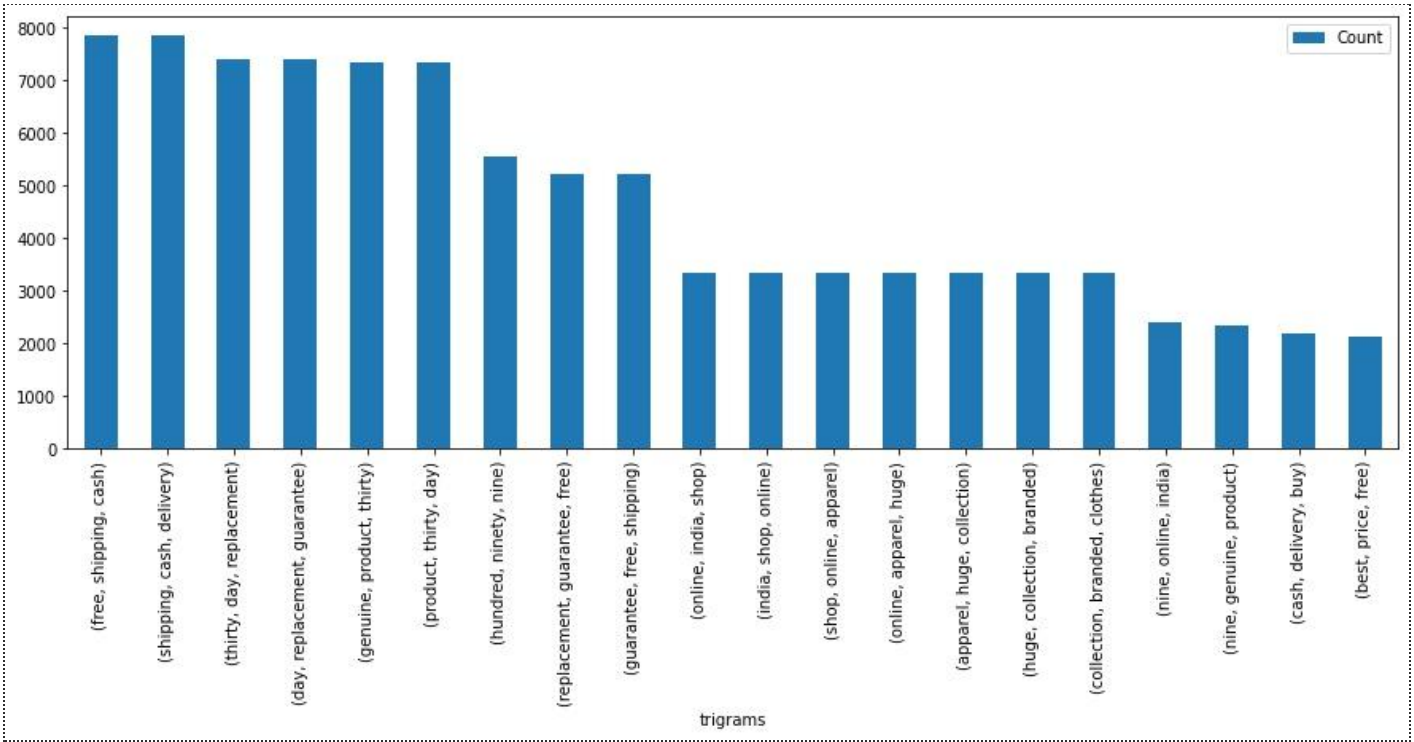
VII. Top 20 Bigrams distribution:



Insights:

- “Guarantee Product” is the bigram having the highest number of occurrences, this means that the many products must have some guarantee.
- “Free shipping” is the bigram having the second-highest number of occurrences, this means that many products have free shipping at home.
- Many items have a “replacement guarantee”.
- “online india”, this bigrams means that all the items can be ordered online in India.
- “Thirty day” and “replacement guarantee”, these both bigrams having almost the same number of occurrences, this means that the products have thirty day replacement guarantee.

VIII. Top 20 Trigrams distribution.



Insights:

- From these two most occurring trigrams, “free shipping cash” and “shipping cash delivery”, I can interpret that there is free shipping cash delivery.
- From these two trigrams, “thirty day replacement” and “day replacement guarantee”, I can interpret that there is a thirty day replacement guarantee.
- “online indian shop” is telling me that this is an online shop platform in India.

11. WEB APPLICATION [Optional]

It was not required in the instructions but I did it so that others can also use it and can test on some test cases. I have created a basic web app and hosted it on Heroku as I also love/enjoy creating web applications in Flask or Django.

Framework Used:

- Flask (for backend)
    - Ref: https://flask.palletsprojects.com/en/1.1.x/
- Tailblocks CSS (for frontend)
    - Ref: https://tailblocks.cc/

Web Application Link:

https://midas-task3-shaneywaris.herokuapp.com/

12. Github Repositories:

Task 3 Code Github Repository:

- https://github.com/ShaneyWaris/MIDAS_Task3_NLP_Code

WebApp Github Repository:

- https://github.com/ShaneyWaris/Midas_Task3_ShaneyWaris

TASK 3 COMPLETED :-)
