REPORT
Natural Language Processing Task

Made by - Kushagra Garg
BTech(IT), 3rd Year
USICT, Delhi
Email - kushagra.garg19@gmail.com
Contact - +91 9811766270

Contents

"AI is likely to be the either the best or the worst thing to happen to humanity" - Stephen Hawking

1. Introduction

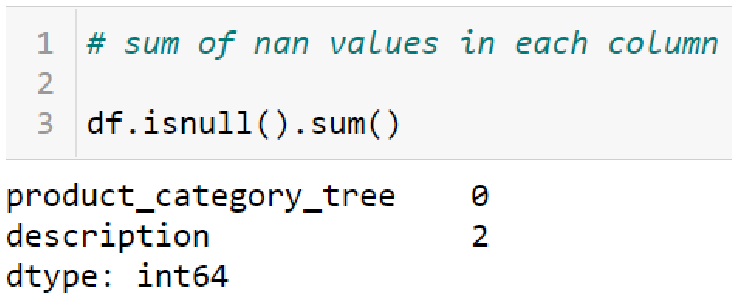
The dataset given to us is a flipkart products listings dataset where each row represents a different product listed on flipkart.com. The columns specify different properties of a product listing (Brand, Description, Price, Url, Image) etc. So, we are asked to predict the primary category for product listing using description as the main feature.

This is a classical text classification problem in natural language processing. In text classification, we preprocess the text by understanding by the frequency and semantics of words and sentences. Then, after cleaning the text, we vectorize it into a form that can be fed to a machine learning or deep learning model. Then, we evaluate the performance of our model based on the predictions.

2. Data Analysis and Visualization

The dataset consists of 20k rows and 15 columns. I separated the description & product_category_tree column as they are features and classes respectively.

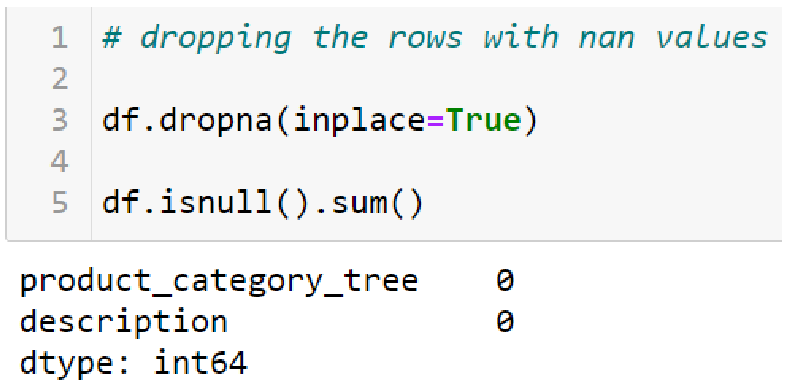
Dropping null values from these columns -



```
# sum of nan values in each column

df.isnull().sum()
```

```
product_category_tree    0
description              2
dtype: int64
```



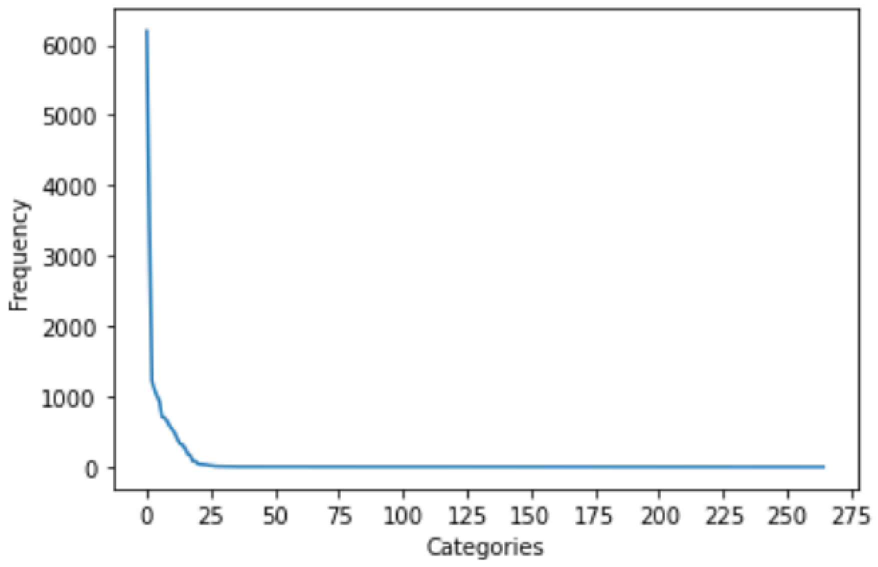
```
# dropping the rows with nan values

df.dropna(inplace=True)

df.isnull().sum()
```

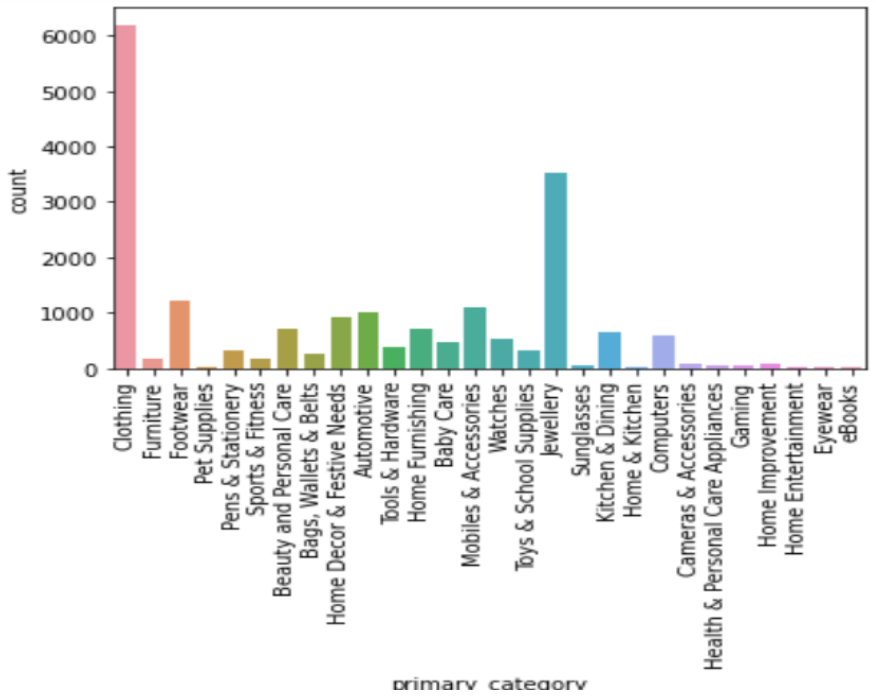
```
product_category_tree    0
description              0
dtype: int64
```

Finding the primary category from the product_category_column and then visualizing the categories by plotting them (decreasing order of their frequency) -



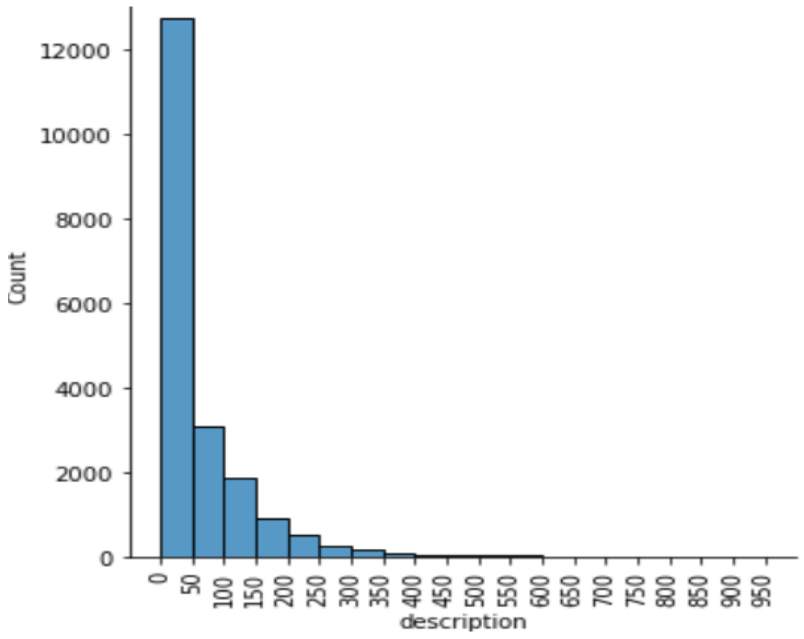
I can observe that the data for categories after index 25 is very less, so I have to remove these categories due to lack of data. Also, I can notice that categories having frequencies less than 10 aren't actual categories, so I considered it the basis of filtering.

Visualizing the final categories using a count plot -



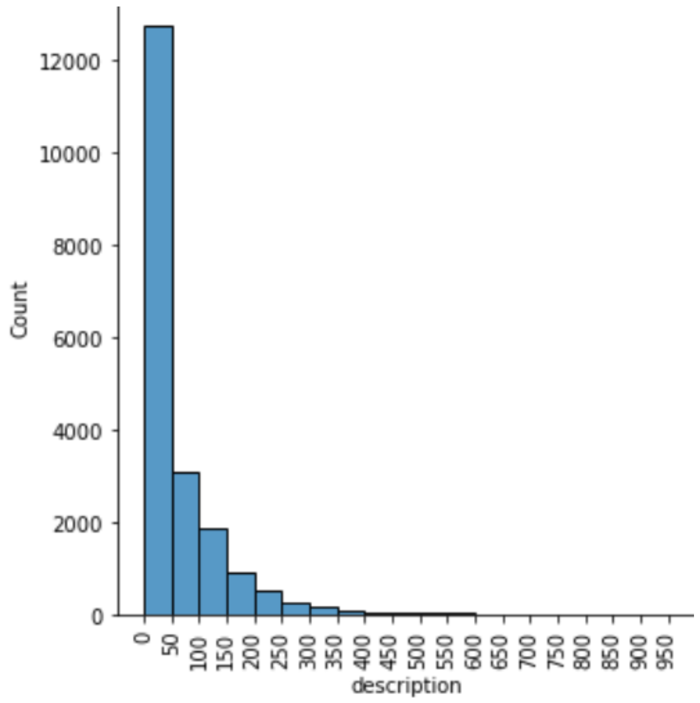
Description Column Analysis -

Plotting the no of characters in each row of description -



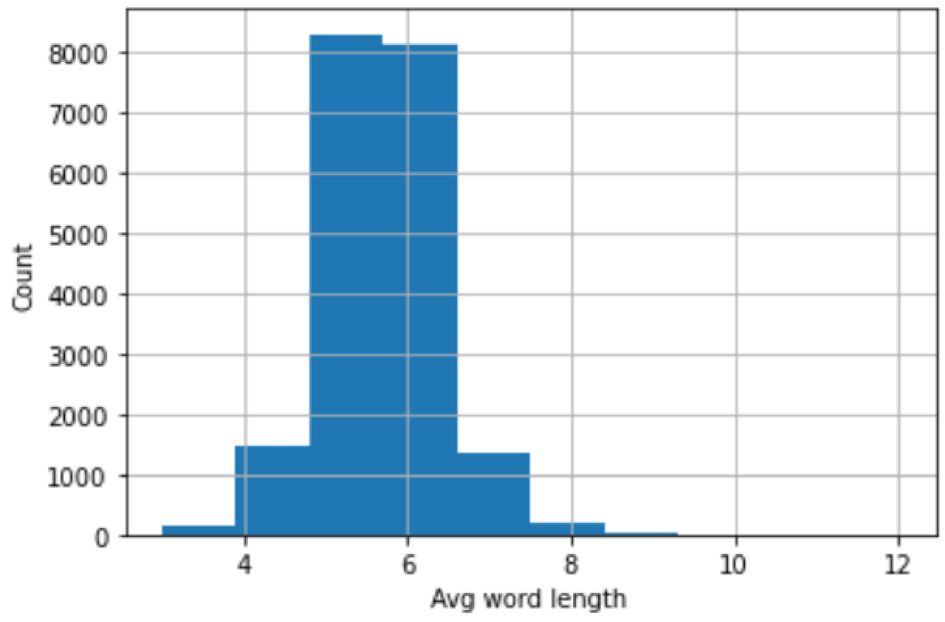
No of characters present in each description lies between 100 and 950.

Plotting the no of words in each row of description -



No of words present in each description lies between 0 and 150.

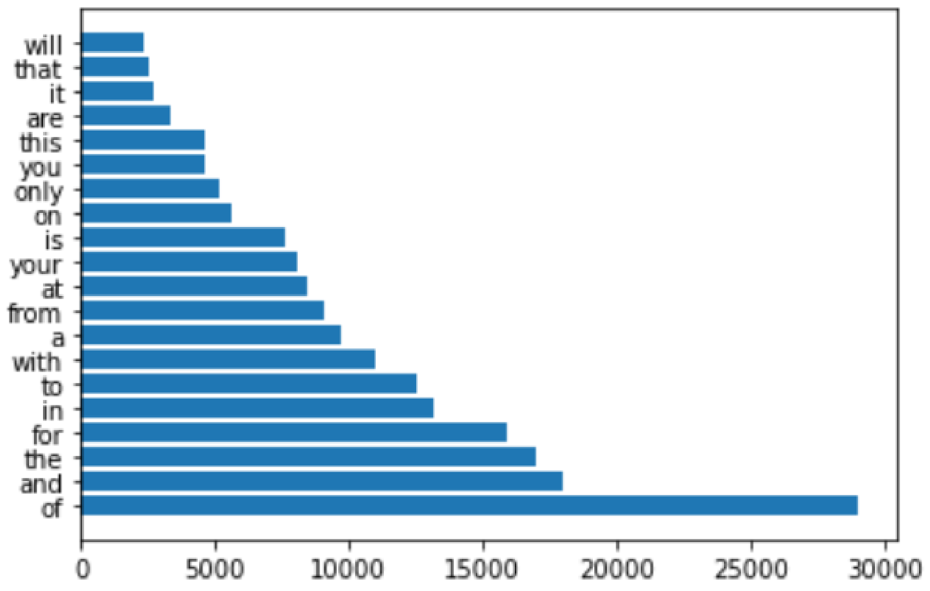
Plotting the average word length in each row of description -



Average word length in each description lies between 4 to 6.

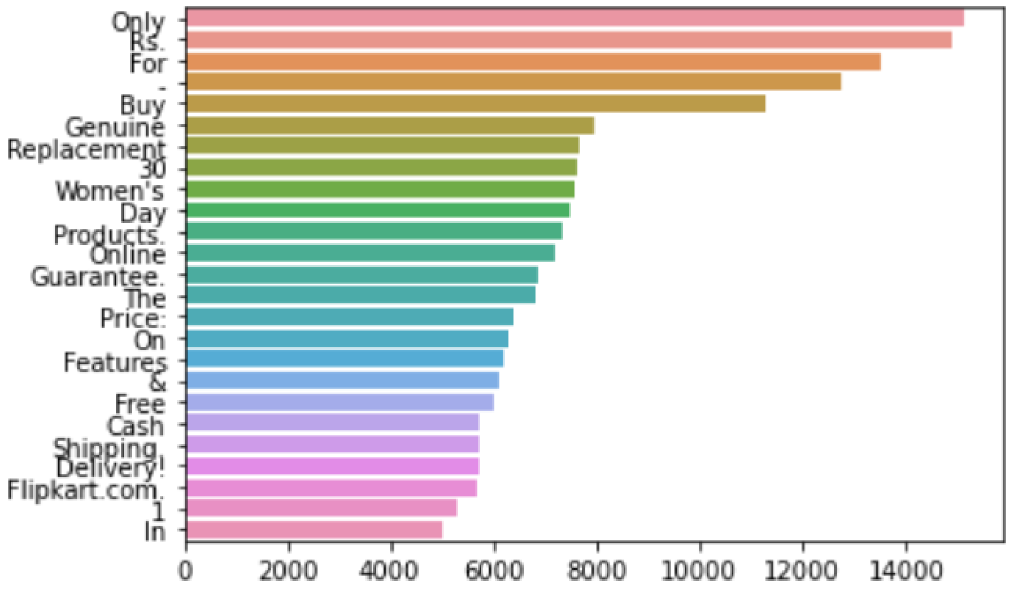
The graph is left skewed due to the no of stopwords in description
Analyzing the amount and the types of stopwords can give us some good insights into the data.

Analysing the no of stopwords by plotting the graph of most frequent stopwords -



Average word length in each description lies between 4 to 6.

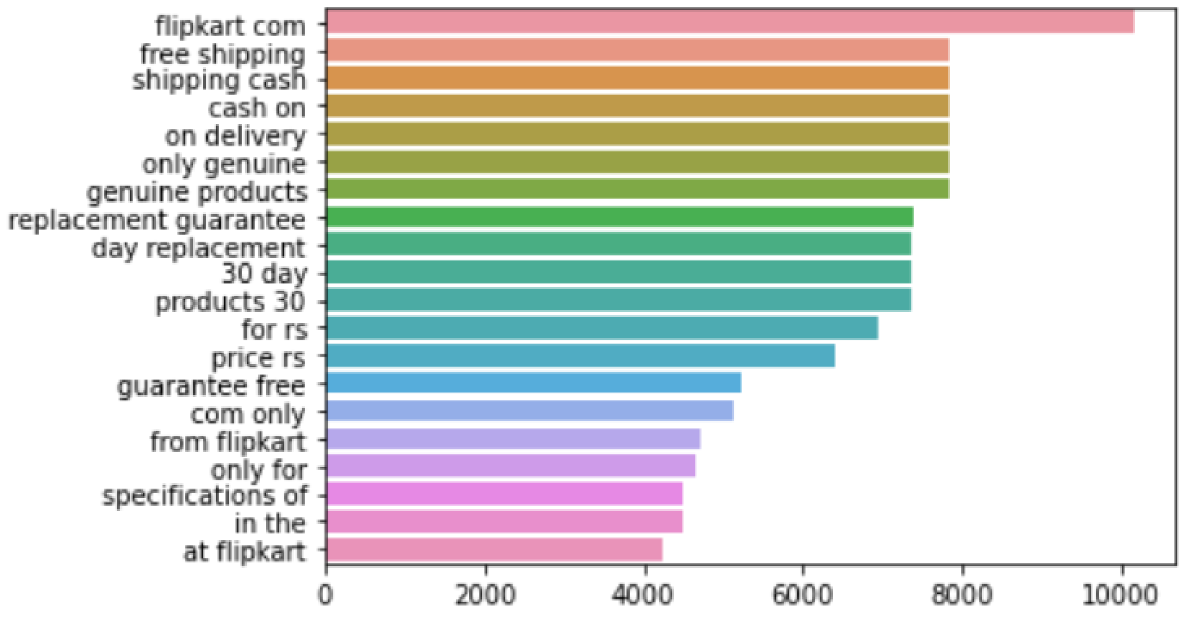
Plotting the most common words appearing in the description -



We can observe that most of the frequent common words (only, Rs, Buy etc.) do not help in predicting the primary category. So, it's better to include them in stopwords.

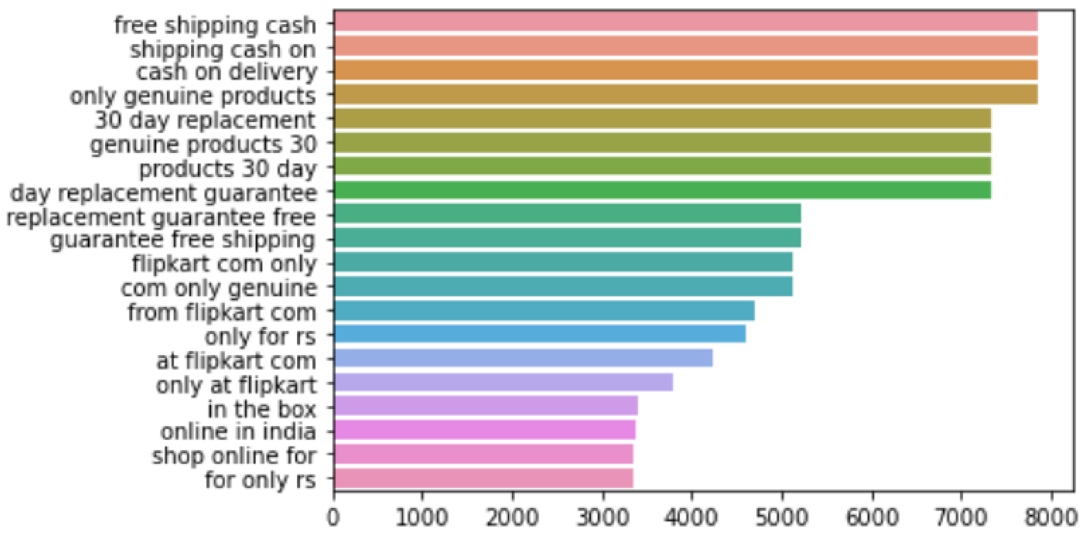
Looking at most frequent n-grams can give you a better understanding of the context -

Plotting the most common bi-grams appearing in the description -



The most common bi-grams are the generic words appearing in every description. So, it's not a good idea to use them as features.

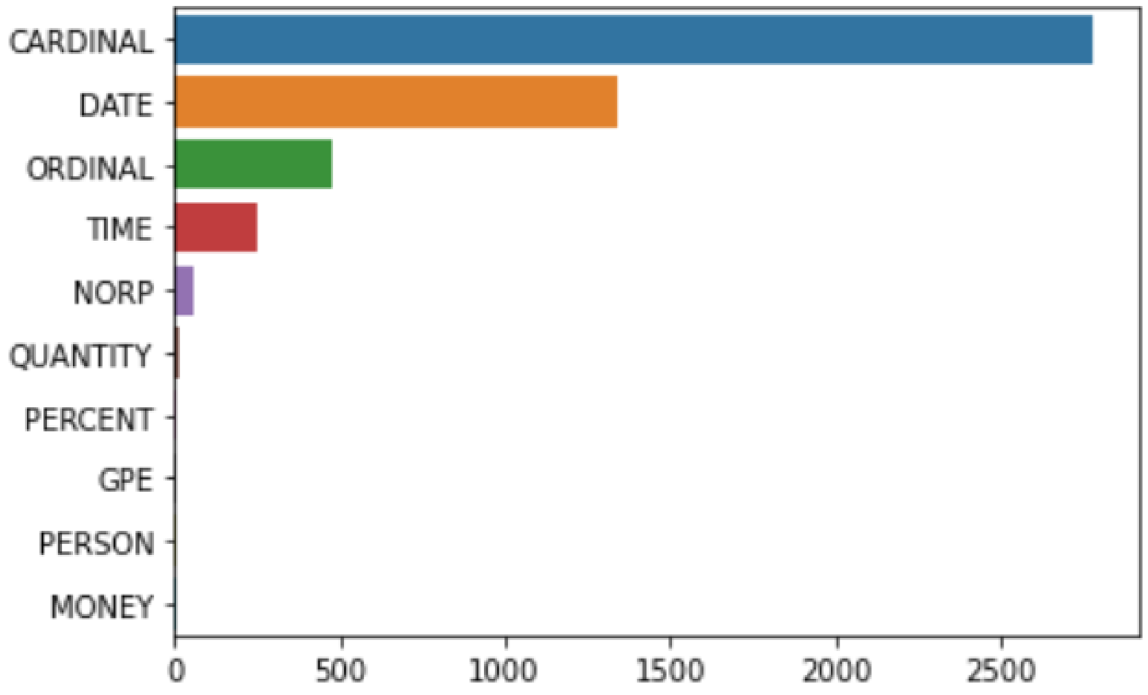
Plotting the most trigrams words appearing in the description -



The most common bi-grams are also the generic words appearing in every description. So, it's not a good idea to use them as features either.

Analyzing the description column using NER

Plotting the most common types of entities in our description -



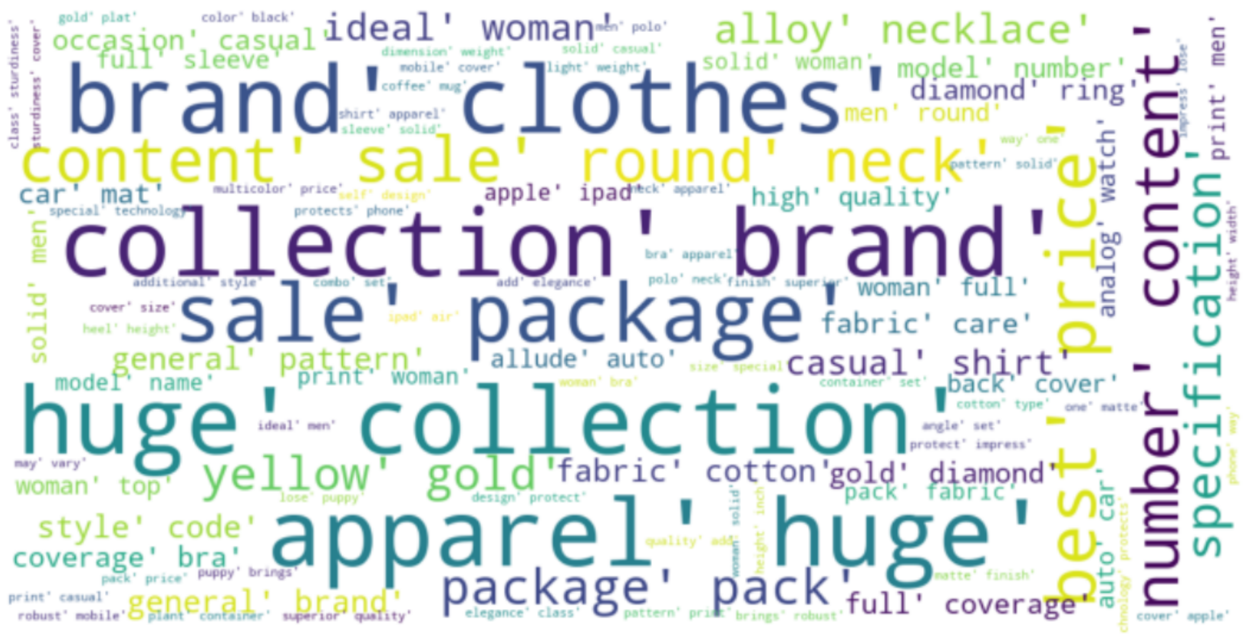
We can observe numbers are cardinals, date, ordinal are appearing often in our data.

3. Data Cleaning and Preprocessing

For data cleaning and preprocessing, we follow the below steps -

1. Reduce every word in the description to its base form using WordNetLemmatizer which uses part of speech to find the root form.
2. Filter the stop words and ordinals.
3. Spell check each word using Speller from autocorrect library.
4. Then, we append each word to the corpus.

Word cloud of most common words after data cleaning -



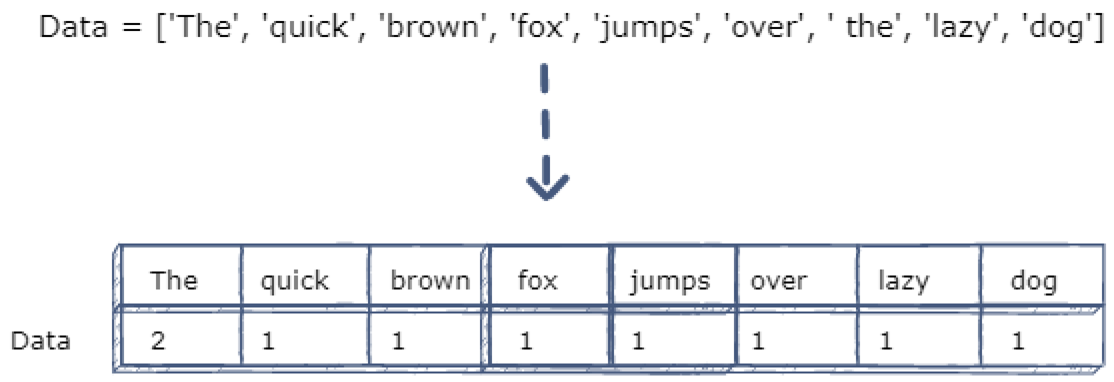
We can observe that our cleaned words have no stopwords and are unique for our categories.

4. Text Vectorization

As we know, text cannot be interpreted by machine models. So, we have to vectorize them, and later feed them into machine learning models.

The techniques used for text vectorization are as follows -

1. **Count Vectorizer** - CountVectorizer tokenizer(tokenization means breaking down a sentence or paragraph or any text into words) the text along with performing very basic preprocessing like removing the punctuation marks, converting all the words to lowercase, etc



```
Data = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', ' the', 'lazy', 'dog']
```

	The	quick	brown	fox	jumps	over	lazy	dog
Data	2	1	1	1	1	1	1	1

2. **TF-IDF Vectorizer** - The TfidfVectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents.

Term Frequency is calculated with the following formula:

$$Term\ Frequency(t) = \frac{number\ of\ times\ t\ appears\ in\ a\ document}{total\ number\ of\ terms\ in\ the\ document}$$

Inverse Document Frequency is calculated with the following formula:

$$IDF(t) = log_e(\frac{Total\ Number\ of\ Documents}{Number\ of\ Documents\ with\ t\ in\ it})$$

5. Machine Learning Models Used

The following factors were taken into consideration while choosing the machine learning models -

1. Models which give high accuracy in text classification
2. High performance models in general (ensemble, stacking models)
3. Models which require less time to fit the training data and produce results.

Thus, the following models were used to predict categories -

1. **Multinomial Naive Bayes** - Naive Bayes classifier for multinomial models. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

2. **Support Vector Machines** - The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

3. **Random Forest** - Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean/average prediction of the individual trees.

4. **K-Nearest Neighbours** - The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slower as the size of that data in use grows.

5. **Gradient Boosting Trees** - When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

6. **XgBoost -** XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. The same code runs on a major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

6. ML models performance

The following accuracies were included in the final results to judge the performance of each model.

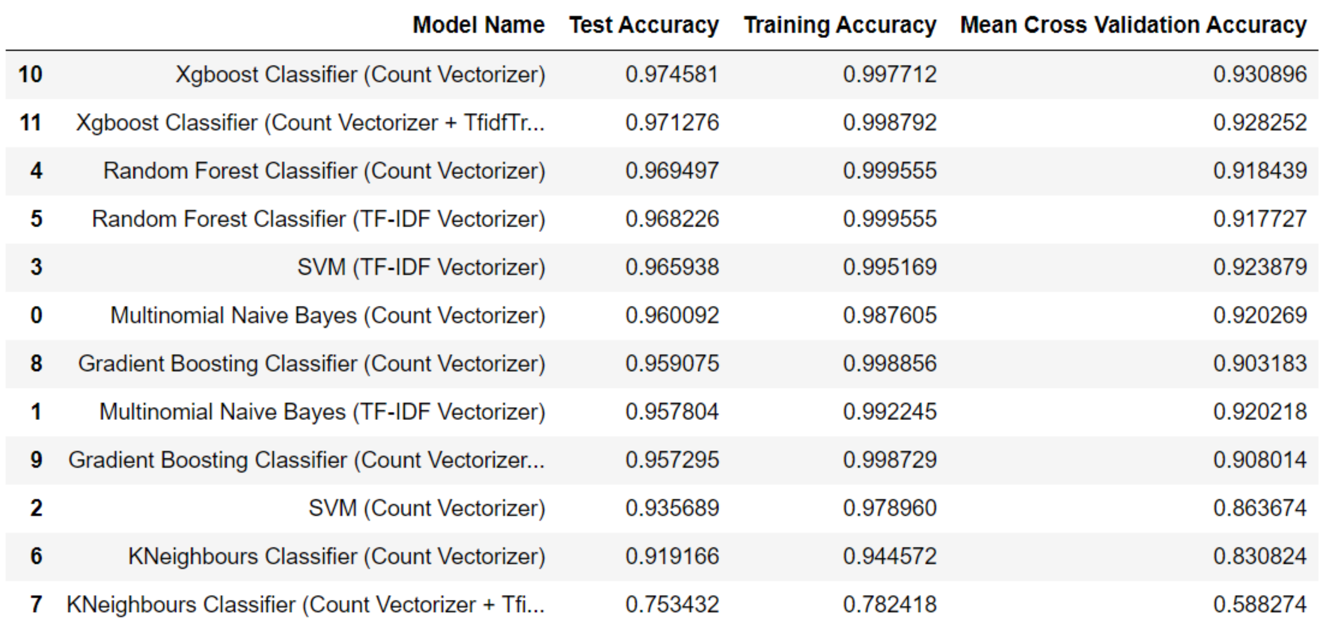
Before feeding data into the classifier, the data was split into training and testing sets using 80:20 ratio, then both the sets were vectorized by Count Vectorizer and TF-IDF Vectorizer and results were recorded for the following accuracies.

1. **Test Accuracy** - The accuracy obtained on the testing set after training the model on the training set. This is calculated to check how our models perform on the data it hasn't seen before.

2. **Training Accuracy** - The accuracy obtained on the training set after the model is trained. This is calculated to check if models underfit or overfit the training data.

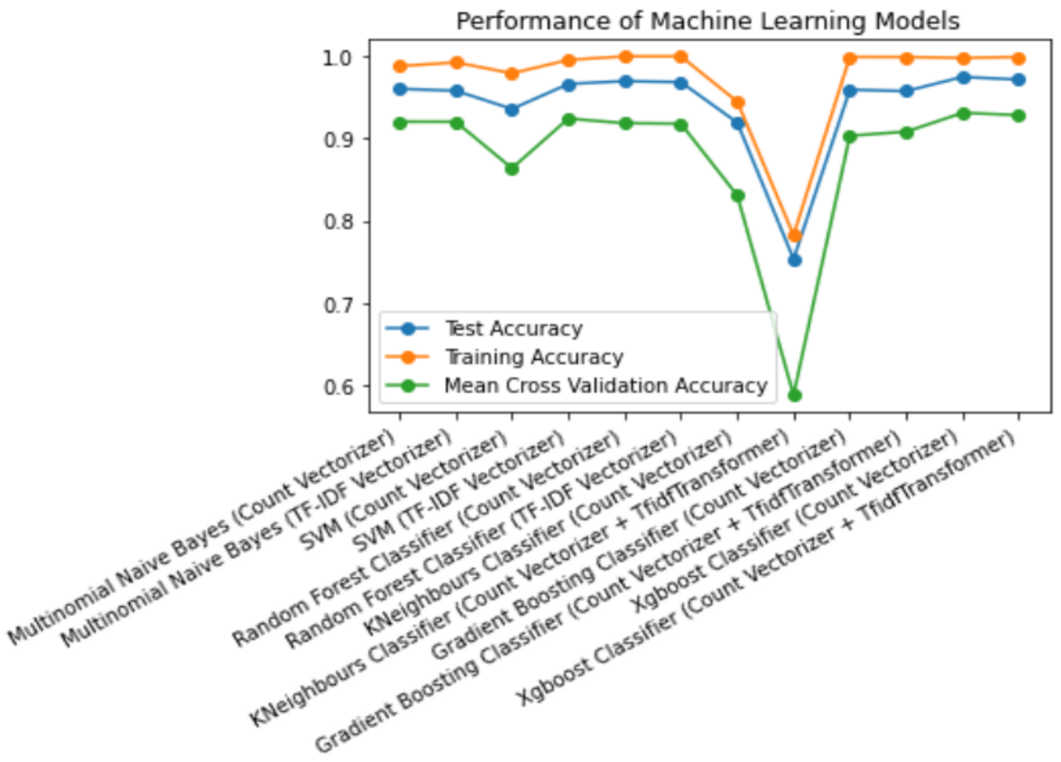
3. **Cross Validation Accuracy** - The mean accuracy of cross validation from 5 K-Fold is obtained from the whole dataset. The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

Table depicting the results of each model along with the vectorizer used (in decreasing order of test accuracy) -



	Model Name	Test Accuracy	Training Accuracy	Mean Cross Validation Accuracy
10	Xgboost Classifier (Count Vectorizer)	0.974581	0.997712	0.930896
11	Xgboost Classifier (Count Vectorizer + TfidfTr...	0.971276	0.998792	0.928252
4	Random Forest Classifier (Count Vectorizer)	0.969497	0.999555	0.918439
5	Random Forest Classifier (TF-IDF Vectorizer)	0.968226	0.999555	0.917727
3	SVM (TF-IDF Vectorizer)	0.965938	0.995169	0.923879
0	Multinomial Naive Bayes (Count Vectorizer)	0.960092	0.987605	0.920269
8	Gradient Boosting Classifier (Count Vectorizer)	0.959075	0.998856	0.903183
1	Multinomial Naive Bayes (TF-IDF Vectorizer)	0.957804	0.992245	0.920218
9	Gradient Boosting Classifier (Count Vectorizer...	0.957295	0.998729	0.908014
2	SVM (Count Vectorizer)	0.935689	0.978960	0.863674
6	KNeighbours Classifier (Count Vectorizer)	0.919166	0.944572	0.830824
7	KNeighbours Classifier (Count Vectorizer + Tfi...	0.753432	0.782418	0.588274

Plotting the accuracy of each each model using line plot -



We can clearly observe that Xgboost classifier using Count Vectorizer outperformed every other model in terms of training accuracy as well as mean cross validation accuracy (suggesting that it hasn't overfitted on training set). Other classifiers that performed good were Random Forest Classifier, SVM, Multinomial Naive Bayes. However, Gradient Boosting classifier suffered overfitting on training set resulting in less mean cross validation accuracy and Random Forest Classifier also suffered this to some extent. K-Neighbours classifier performed poorly of all the classifiers used.

7. Deep Learning Approach

The exponential growth in the number of complex datasets every year requires more enhancement in machine learning methods to provide robust and accurate data classification. Lately, deep learning approaches are achieving better results compared to previous machine learning algorithms on tasks like image classification, natural language processing, face recognition, etc. The success of these deep learning algorithms relies on their capacity to model complex and non-linear relationships within the data.

Deep Neural Network - Deep Neural Networks architectures are designed to learn through multiple connections of layers where every single layer only receives a connection from previous and provides connections only to the next layer in the hidden part.

The input layer is embedding vectors. The output layer neurons equal to the number of classes for multi-class classification and only one neuron for binary classification. Here, we have multi-class DNNs where the number of nodes in each layer as well as the number of layers are randomly assigned.

The implementation of Deep Neural Network (DNN) is basically a discriminatively trained model that uses the standard back-propagation algorithm and sigmoid or ReLU as activation functions. The output layer for multi-class classification should use Softmax.

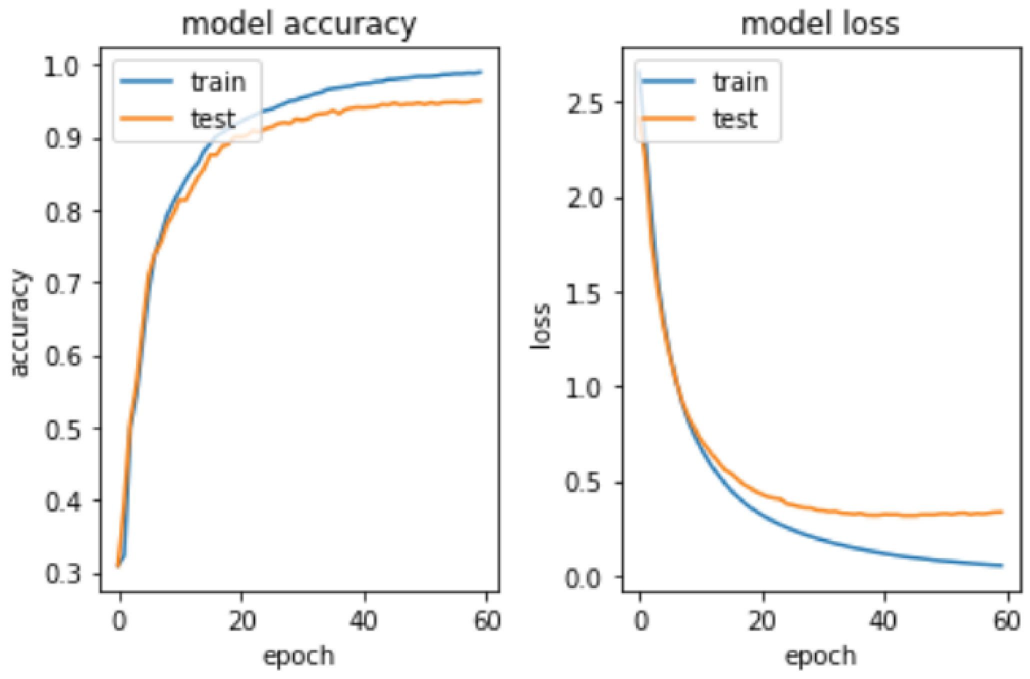
8. Deep Neural Networks Performance (DNN)

The deep neural network performance is evaluated on the following parameters -

1. **Test Accuracy** - The accuracy obtained on the testing set after training the model on the training set. This is calculated to check how our models perform on the data it hasn't seen before.

2. **Training Accuracy** - The accuracy obtained on the training set after the model is trained. This is calculated to check if models underfit or overfit the training data.

Plotting the accuracy per epoch for DNN with Words Embeddings -



Plotting the accuracy per epoch for DNN with Count Vectorizer -

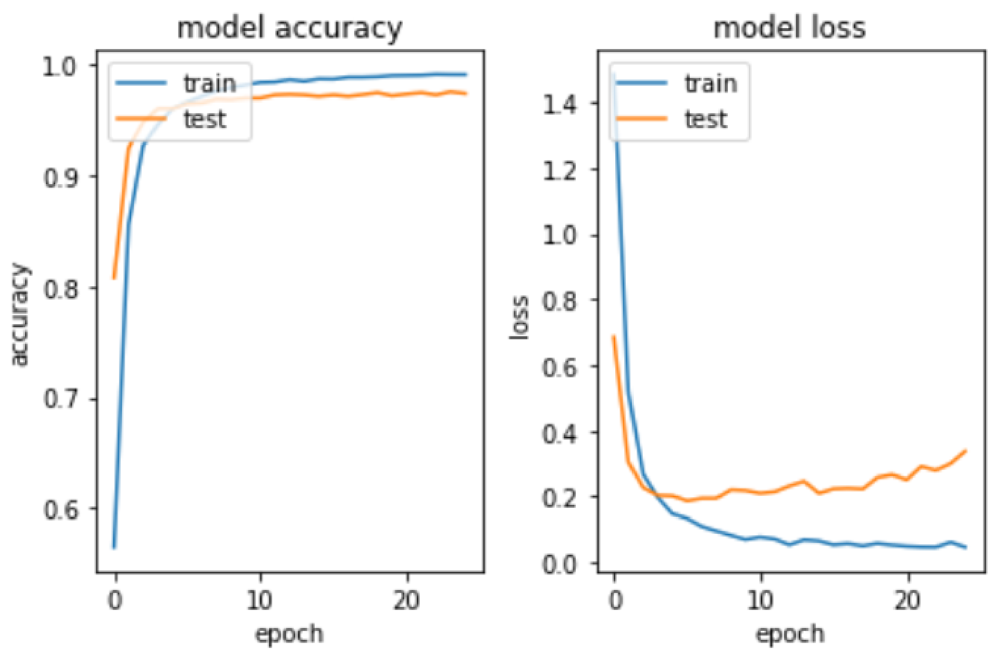
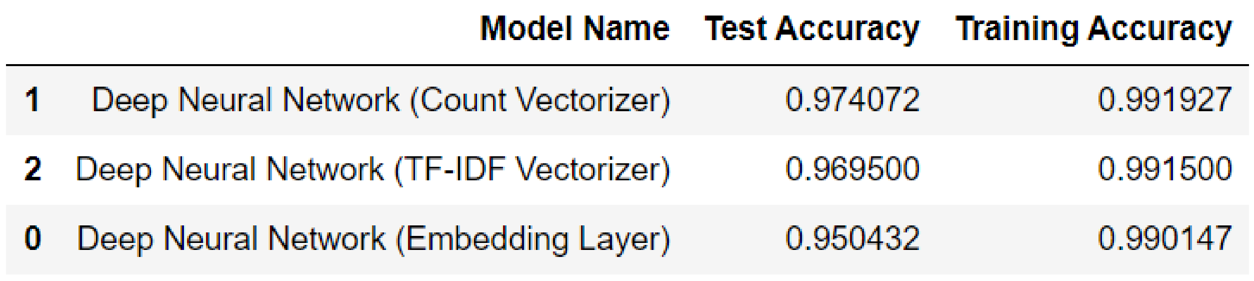
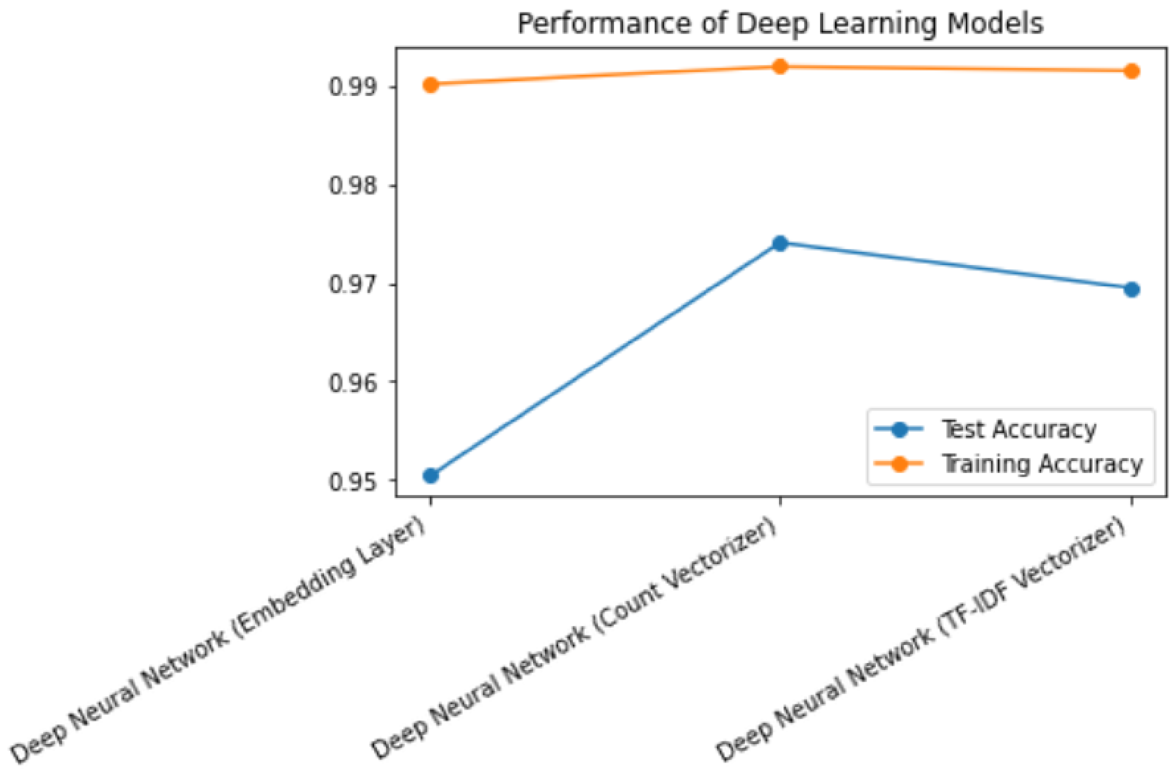


Table depicting the results of each deep learning model along with the vectorizer used (in decreasing order of test accuracy) -



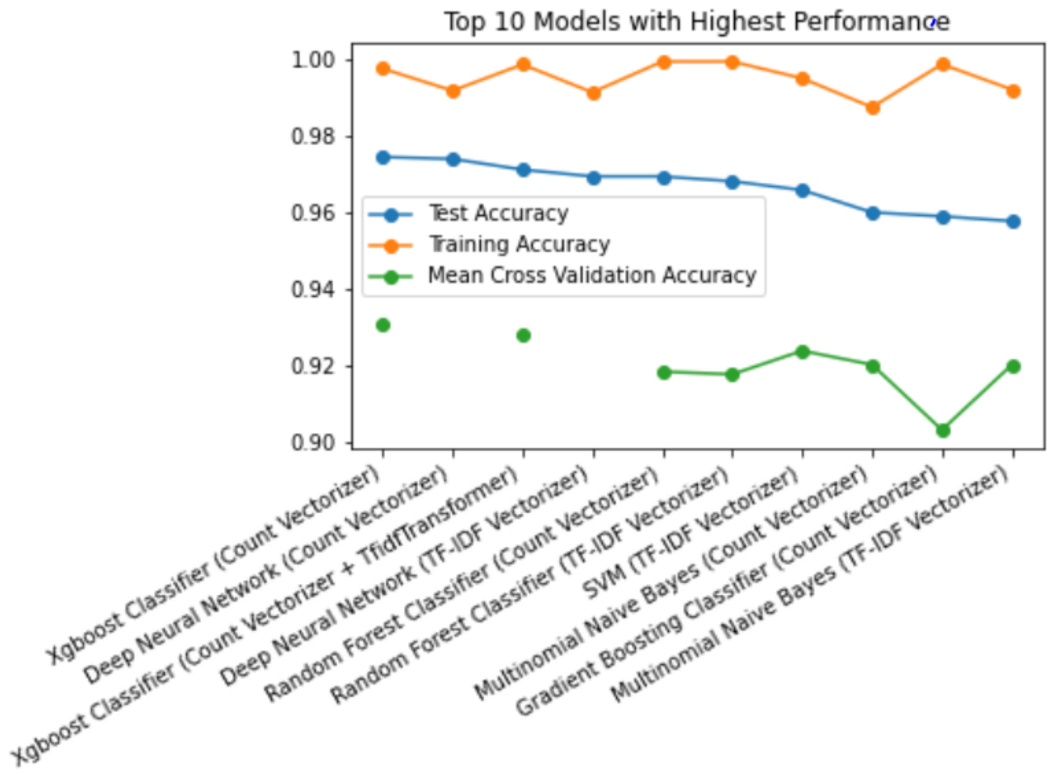
	Model Name	Test Accuracy	Training Accuracy
1	Deep Neural Network (Count Vectorizer)	0.974072	0.991927
2	Deep Neural Network (TF-IDF Vectorizer)	0.969500	0.991500
0	Deep Neural Network (Embedding Layer)	0.950432	0.990147

Plotting the accuracy of each each deep learning model using line plot -



We can observe that deep neural network layers performed really well compared to traditional machine learning models such as K-Neighbours, SVM, Multinomial Naive Bayes. Although there is a chance of overfitting in these models, that can be adjusted by fine tuning the hyperparameters. Among the three deep neural network variations implemented by changing the input type, the one implemented with Count Vectorizer gave the most promising results.

9. Conclusion



Xgboost Classifier (Count Vectorizer) and Deep Neural Networks (with Count Vectorizer) outperformed all other models and post fine tuning of hyperparameters and training with more data), they can be used to predict primary categories on the flipkart products listing to get the most accurate results.

10. Citations

1. https://neptune.ai/blog/text-classification-tips-and-tricks-kaggle-competitions
2. https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools
3. https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python/
4. https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/
5. https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/
6. https://pub.towardsai.net/natural-language-processing-nlp-with-python-tutorial-for-beginners-1f54e610a1a0
7. https://www.machinelearningplus.com/nlp/natural-language-processing-guide/
8. https://towardsdatascience.com/ml-powered-product-categorization-for-smart-shopping-options-8f10d78e3294
9. https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/
10. https://medium.com/text-classification-algorithms/text-classification-algorithms-a-survey-a215b7ab7e2d
11. https://machinelearningmastery.com/xgboost-for-imbalanced-classification/#:~:text=The%20XGBoost%20algorithm%20is%20effective,over%20the%20model%20training%20procedure.
12. https://medium.datadriveninvestor.com/deep-learning-techniques-for-text-classification-9392ca9492c7