Task 3

(NLP)

Data Cleaning & Analysis:

From the original dataset, the following columns were dropped. As stated in the problem statement, description column is the main feature, hence I dropped the columns with null description values. The code snippet for dropping the columns is shown below along with the columns left for the problem.

```
In [4]: dataset = pd.read_excel("flipkart_com-ecommerce_sample - flipkart_com-ecommerce_sample.xlsx", engine="openpyxl")
        dataset = dataset.drop(columns=["uniq_id", "crawl_timestamp", "image", "product_url", "pid", "product_specifications"])
        dataset = dataset[dataset["description"].notna()]
        dataset.head()
```

Out[4]:

	product_name	product_category_tree	retail_price	discounted_price	is_FK_Advantage_product	description	product_rating	overall_rating	brand
0	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	999.0	379.0	False	Key Features of Alisha Solid Women's Cycling S...	No rating available	No rating available	Alisha
1	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	32157.0	22646.0	False	FabHomeDecor Fabric Double Sofa Bed (Finish Co...	No rating available	No rating available	FabHomeDecor
2	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	999.0	499.0	False	Key Features of AW Bellies Sandals Wedges Heel...	No rating available	No rating available	AW
3	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	699.0	267.0	False	Key Features of Alisha Solid Women's Cycling S...	No rating available	No rating available	Alisha
4	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	220.0	210.0	False	Specifications of Sicons All Purpose Arnica Do...	No rating available	No rating available	Sicons

Few columns had most of the values as null. Hence, these below columns were dropped.

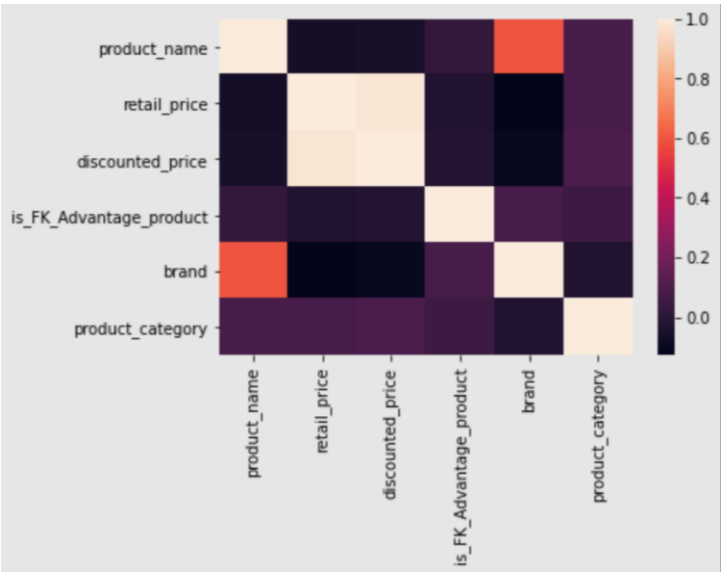
```
6   product_rating          1849 non-null   float64
7   overall_rating          1849 non-null   float64
```

Some of the other column values were also null in the given dataset (as shown below). Instead of dropping such columns, I used **KNNImputer** to fill in the missing values. Also, to convert the string values into integral values, **Label Encoder** was used.

```
In [5]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19998 entries, 0 to 19999
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   product_name            19998 non-null  object
 1   product_category_tree   19998 non-null  object
 2   retail_price            19920 non-null  float64
 3   discounted_price        19920 non-null  float64
 4   is_FK_Advantage_product 19998 non-null  bool
 5   description             19998 non-null  object
 6   product_rating          19998 non-null  object
 7   overall_rating          19998 non-null  object
 8   brand                   14135 non-null  object
dtypes: bool(1), float64(2), object(6)
memory usage: 1.4+ MB
```

A correlation matrix was created for the columns left in the dataset. In the following output it can be observed that discounted_price and retail_price are highly correlated and hence only one of them should be used as a parameter. Similarly, brand and product_name are also highly correlated.



Hence the columns finally used as input parameters are: Product Description, Product Name, Retail Price and isFkAdvantageProduct.

In the dataset, we are provided with Product Category Tree instead of actual Product Category. The product category column is created by extracting the root of the product category tree.

E.g. Product category tree is:

```
Baby Care >> Baby & Kids Gifts >> Stickers >> Uberlyfe Stickers,
```

Then the product category will be “Baby Care”

As there were a few categories with less data, all the categories having data points less than 100 were removed from the dataset to balance the dataset.

Once the Product Category column is created, label encoding is done to convert string to integer.

[Note: Model was also trained with all the categories. Results are shown in the table below]

"Product Description" Preprocessing and Encoding:

Each product description was tokenized using NLTK's TweetTokenizer. From each token, stop words were removed, if any. After filtering out the stop words, the necessary words were lemmatized using NLTK's WordNetLemmatizer. The lemmatized words were then added to the Bag of Words. Each word was provided a unique ID which is used while training the model. The length of the product description used for input dataset was set to 430 (mean of the lengths of product description), i.e. if a product description has number of words (after lemmatization) greater than 430 only the latest 430 words are considered as input, and if number of words less than 430, then the input sequence is padded with 0's.

Experiments:

Same model architecture with different number of parameters were trained during the research. The models were trained with max input length of description as 80 and 430. These models were trained and compared on same train and test split.

Results are shown below:

With all product categories

Model	Embed Size	LSTM Units	Train Accuracy		Test Accuracy	
			80 (%)	430 (%)	80 (%)	430 (%)
Model-1	16	128	**97.309**	97.269	**89.528**	89.508
Model-2	16	256	97.339	**97.659**	89.458	**91.559**
Model-3	16	512	**97.399**	94.399	**90.359**	78.287
Model-4	64	128	97.859	**98.049**	92.039	**93.159**
Model-5	64	256	97.959	**98.309**	92.779	**93.999**
Model-6	64	512	98.239	**98.409**	93.759	**94.419**

With filtered product categories

Model	Embed Size	LSTM Units	Train Accuracy	Test Accuracy
Model-1	32	128	98.589	93.560
Model-2	32	256	98.931	95.095
Model-3	32	512	99.077	95.987
Model-4	64	128	98.859	95.105
Model-5	64	256	99.056	95.686
Model-6	64	512	**99.211**	**96.515**

Future Experiments:

The columns “product_specifications” and “image” can play an important role in improving the model accuracy. The column “product_specifications” contains multiple other features of the product. A CNN approach can be applied to the images of the product. Combining the CNN model and the RNN (proposed in this solution) can give more accurate results.