

Text Classification Project

Aim: To predict the product category from the product description for an e-commerce website.

Programming language used: Python 3.7.4 (Anaconda)

Development environment used: Jupyter Notebook

Python libraries used

Pandas:

pandas is a powerful and flexible open source library for data analysis and manipulation. It offers data structures: DataFrames and Series that represent data in a tabular form, and also provides methods to analyse, filter and manipulate the data. It is imported with the standard alias pd.

Matplotlib:

Matplotlib is a powerful data visualization library that allows us to draw graphs and charts from the data in a pandas dataframe. The pyplot module of the library is the most useful library for the plotting of graphs and is imported with the standard alias plt.

Seaborn:

seaborn is a data visualization tool based on matplotlib. It provides high level interface for drawing attractive and informative graphs. It is imported with the alias sns.

Scikit-learn (sklearn):

Scikit-learn is a powerful library that is commonly used to build machine learning models. It supports classification, regression, and clustering algorithms. It also provides tools for data preprocessing and evaluation metrics for a machine learning model.

Functionalities of scikit-learn used

sklearn.model_selection.train_test_split

It is used to split the dataset into train and test sets. The train set is used to train the model. The model is fit on the train set. The test set is used for testing the model after training.

sklearn.feature_extraction.text.CountVectorizer

CountVectorizer is used to transform a given text into a vector on the basis of the frequency of each word in the text. CountVectorizer returns a matrix in which the columns contain each unique word in the document. The rows contain the text samples. The values in the cells give the frequency of occurrence of the word in that text sample.

sklearn.naive_bayes.MultinomialNB

MultinomialNB allows us to implement the Multinomial Naïve Bayes algorithm for classification; which is a popular algorithm used for text classification models.

sklearn.preprocessing.LabelEncoder

LabelEncoder is used for numerically encoding the target feature. It encodes the target feature with values between 0 to (n-1) where n is the number of classes.

sklearn.metrics.classification_report

It returns a text report showing the main classification metrics like accuracy, recall, and f1_score.

Dataset used

The dataset contains the details of 20000 products sold on the Indian e-commerce giant: Flipkart.

Number of rows and columns: 20000 rows and 15 columns

Names of the columns:

- uniq_id
- crawl_timestamp
- product_url
- product_name
- product_category_tree
- pid
- retail_price
- discounted_price
- image
- is_FK_advantage_product
- description
- product_rating
- overall_rating
- brand
- product_specifications

Target variable: product_category_tree

Feature used to predict the target: description

Link to the dataset:

<https://docs.google.com/spreadsheets/d/1pLv0fNE4WHokpJHUIs-FTVnmI9STgog05e658qEON0I/edit#gid=1396401268>

APPROACH

Importing the required libraries

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
```

Reading the data

Importing the data from an excel sheet and storing it in a pandas DataFrame.

The pandas.DataFrame.read_excel method imports the data from an excel sheet and creates a pandas DataFrame containing the data. It takes the name of the excel workbook as an argument.

```
data = pd.read_excel("ecommerce.xlsx")
```

Viewing the first five rows and the dimensions of the DataFrame

The pandas.DataFrame.head() method is used to view the first five rows, and the pandas.DataFrame.shape attribute is used to view the dimensions of the dataframe.

```
1 data.head()
2
3 # Viewing the first 5 rows
```

	uniq_id	crawl_timestamp	product_url	product_name	product_category_tree	pid	retail
0	c2d766ca982eca8304150849735ffe9	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	SRTEH2FF9KEDEFGF	
1	7f7036a6d550aaa89d34c77bd39a5e48	2016-03-25 22:59:23 +0000	http://www.flipkart.com/fabhomedecor-fabric-do...	FabHomeDecor Fabric Double Sofa Bed	["Furniture >> Living Room Furniture >> Sofa B...	SBEEH3QGU7MFYJFY	3
2	f449ec65dcbc041b6ae5e6a32717d01b	2016-03-25 22:59:23 +0000	http://www.flipkart.com/aw-bellies/p/itmeh4grg...	AW Bellies	["Footwear >> Women's Footwear >> Ballerinas >...	SHOEH4GRSUBJGZXE	
3	0973b37acd0c664e3de26e97e5571454	2016-03-25 22:59:23 +0000	http://www.flipkart.com/alisha-solid-women-s-c...	Alisha Solid Women's Cycling Shorts	["Clothing >> Women's Clothing >> Lingerie, Sl...	SRTEH2F6HUZMQ6SJ	
4	bc940ea42ee6bef5ac7cea3fb5cfbee7	2016-03-25 22:59:23 +0000	http://www.flipkart.com/sicons-all-purpose-am...	Sicons All Purpose Arnica Dog Shampoo	["Pet Supplies >> Grooming >> Skin & Coat Care...	PSOEH3ZYDMSYARJ5	

```
1 data.shape
2
3 # Looking at the number of rows and columns
(20000, 15)
```

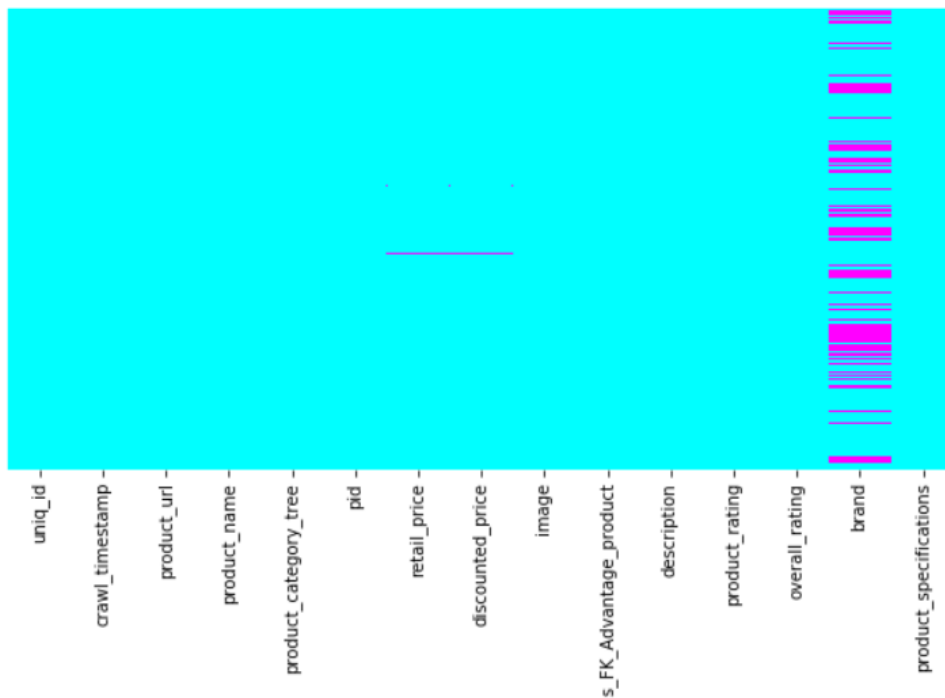
Cleaning and preparing the data

Visualizing missing data in each column

To visualise the number of missing values in each column, a heatmap is drawn using the seaborn library. The `pandas.DataFrame.isnull()` method returns an `DataFrame` of booleans (True if the data in a particular cell is missing and False otherwise). This dataframe is passed as an argument to the `seaborn.heatmap()` method and a heatmap is generated.

From the heatmap, it is clear that the 'brand' feature has a hefty number of values missing.

```
4 plt.figure(figsize = (10,5))
5
6 sns.heatmap(data.isnull(), cmap = "cool", yticklabels = False, cbar = False )
7
8 plt.show()
9
10
```



Checking for duplicate values

```
1 # Checking for duplicate data
2
3 data.duplicated().value_counts()
```

```
False    20000
dtype: int64
```

From the result, it was clear that none of the columns have duplicate values.

Separating the primary category from the category tree

The list 'category' stores the primary categories of all the items. At first, 'category' is declared as an empty list. A for loop iterates over the values in 'product_category_tree' column. The variable 'primary_category' stores the primary category from the product_category_tree column for each row. The inner for loop iterates over the entry of product_category_tree for that particular row. To eliminate non-alphabets, .isalpha() method is used as the condition of the first if statement. The break statement is used to transfer the control of the program out of the inner for loop when the ">" character is encountered. This is because, in the entries of the product_category_tree column, the primary and secondary categories are separated by ">". The words before ">" will be considered as the primary category and will be appended to the list 'category'. Finally, the category list is assigned to the product_category_tree column.

```
1 category = []
2 for element in data.product_category_tree:
3     primary_category = ""
4     for character in element:
5         if (character.isalpha()):
6             primary_category += character
7         if (character == ">"):
8             break
9     category.append(primary_category)
10
11 # print(category)
12
13 data.product_category_tree = category
14
```

Removing redundant categories

Some categories are very similar and they mean exactly the same thing. For example, 'HomeFurnishing' and 'Furniture' mean the same thing. Hence, they should belong to the same category. The following code assigns the category 'Furniture' to 'HomeFurnishing', 'KitchenDining' to 'HomeKitchen', 'BeautyandPersonalCare' to 'HealthPersonalCareAppliances', 'Eyewear' to 'Sunglasses'.

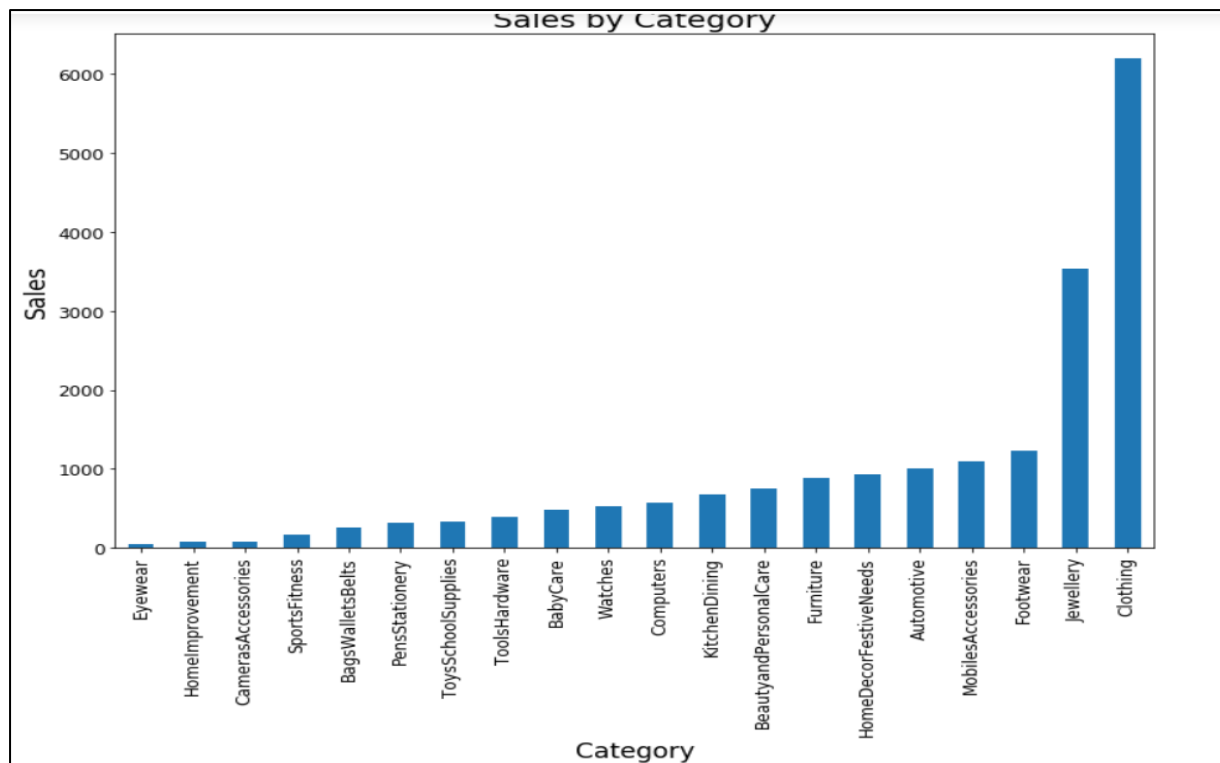
```
for i in range(len(data.product_category_tree)):
    if (data.product_category_tree[i] == "HomeFurnishing"):
        data.product_category_tree[i] = "Furniture"
    elif (data.product_category_tree[i] == "HomeKitchen"):
        data.product_category_tree[i] = "KitchenDining"
    elif (data.product_category_tree[i] == "HealthPersonalCareAppliances"):
        data.product_category_tree[i] = "BeautyandPersonalCare"
    elif (data.product_category_tree[i] == "Sunglasses"):
        data.product_category_tree[i] = "Eyewear"
```

Visualizing the number of sales in the top 20 categories

A bar plot is generated using the matplotlib library that illustrates the number of sales in the top 20 categories.

```
# To visualize the number of items sold in each category

plt.figure(figsize=(12,8))
data.product_category_tree.value_counts()[:20].sort_values(ascending=True).plot(kind='bar')
plt.title("Sales by Category",fontsize=20)
plt.xlabel("Category", fontsize = 17)
plt.ylabel("Sales", fontsize=17)
plt.yticks(fontsize=12)
plt.xticks(fontsize=12)
plt.show()
```



Creating a new DataFrame with only the relevant data

Only those rows are being considered that belong to the top 20 categories.

This is because the rest of the categories are very low in number. The model will not learn anything from the description of those categories, and it might lead to overtraining. Description will be used as the main feature for the model. The following code creates a new dataframe with only the relevant data.

'categories_to_consider' is a list that stores the entries from the 'product_category_tree' that belong to the top 20 categories and hence, will be considered. 'df' is the new dataframe with only the relevant data.

```
categories_to_consider = list(data.groupby('product_category_tree').count().sort_values(by='uniq_id',ascending=False).head(20))
df = data[data.product_category_tree.isin(categories_to_consider)]
df = df[['product_category_tree', 'description']]
```


Encoding the target variable using LabelEncoder

encoder is an object of LabelEncoder class. When the fit_transform method of the encoder object is invoked with the target variable as argument, it returns the encoded values; which is then assigned to the original column in the DataFrame.

```
encoder = LabelEncoder()

encoded_categories = encoder.fit_transform(df['product_category_tree'])

df['product_category_tree'] = encoded_categories
```

Implementing the model

Splitting the dataset into train and test sets

Test set is 20% of the dataset while train set is 80% of the dataset.

```
train_x, test_x, train_y, test_y = train_test_split(df.description.fillna(''), df.product_category_tree, test_size = 0.2)
```

Converting the training data to the vectorized form

The parameter stop_words of the CountVectorizer class is set to "english". This is done so that the model ignores irrelevant and commonly used words in English language that do not contribute to the data processing. 'vectorizer' is the object of the class CountVectorizer. Its fit_transform method returns the vectorized form of the training data.

```
vectorizer = CountVectorizer(stop_words = "english")

X_train_matrix = vectorizer.fit_transform(train_x)
```

Fitting and scoring the training data

‘model’ is an object of the MultinomialNB class. model.fit() method is applied on the vectorized form of the data to fit the model. model.score() method returns the accuracy score of the training data.

```
3
4 model = MultinomialNB()
5
6 model.fit(X_train_matrix, train_y)
7
8 print(model.score(X_train_matrix, train_y))
```

0.9501117853720856

Scoring the test data

```
3 X_test_matrix = vectorizer.transform(test_x)
4
5
6 print(model.score(X_test_matrix, test_y))
7
8 predicted_result = model.predict(X_test_matrix)
9 print(classification_report(test_y, predicted_result))
10
```

0.9315278487480838

	precision	recall	f1-score	support
0	0.94	0.99	0.97	212
1	0.92	0.65	0.76	92
2	0.79	0.61	0.69	49
3	0.86	0.94	0.90	131
4	1.00	0.33	0.50	18
5	0.98	0.99	0.99	1245
6	0.93	0.80	0.86	108
7	1.00	1.00	1.00	10
8	0.97	0.92	0.94	255
9	0.89	0.97	0.93	176
10	0.90	0.98	0.94	193
11	1.00	0.26	0.42	19
12	0.86	1.00	0.92	714
13	0.96	0.92	0.94	128
14	0.98	0.93	0.95	199
15	0.95	0.47	0.63	76
16	1.00	0.40	0.57	40
17	0.95	0.93	0.94	96
18	0.84	0.73	0.78	63
19	0.96	0.98	0.97	90
accuracy			0.93	3914
macro avg	0.93	0.79	0.83	3914
weighted avg	0.93	0.93	0.93	3914

Testing the model on different types of inputs

The following code block tests how the model performs on an input which is slightly different from that in the dataset.

```
1 print(encoder.inverse_transform(model.predict(vectorizer.transform(["Women's ethnic sandals. Material: Leather."])))
2 print(encoder.inverse_transform(model.predict(vectorizer.transform(["Full-sleeve embroidered women's top"]))))
3 print(encoder.inverse_transform(model.predict(vectorizer.transform(["Fabric double sofa bed"]))))
4 print(encoder.inverse_transform(model.predict(vectorizer.transform(["diamond earring gold plated"]))))
5 print(encoder.inverse_transform(model.predict(vectorizer.transform(["Pizza slicer"]))))
6 print(encoder.inverse_transform(model.predict(vectorizer.transform(["Peacock With Meena Art Showpiece"]))))
7 print(encoder.inverse_transform(model.predict(vectorizer.transform(["Cricket bat"]))))
8
9 # Testing the model on different types of inputs
10
```

```
['Footwear']
['Clothing']
['Furniture']
['Jewellery']
['KitchenDining']
['HomeDecorFestiveNeeds']
['SportsFitness']
```

The model could predict the categories correctly so far.

Conclusion

The model has an overall accuracy of 93% on the test data and 95% on the training data.

Some ideas to improve the accuracy of the model:

1. Using a more exhaustive list of stop-words in the testing data.
2. Performing lemmatization: The process of normalizing different grammatical variations of a word to its root form.

In this model, the vectorizer used is CountVectorizer. The same could have been done using TfidfVectorizer as well.

Other algorithms that can be implemented: Support Vector Classifier, Decision Tree Classifier.

References

1. <https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/>
2. <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>
3. https://docs.w3cub.com/scikit_learn/modules/generated/sklearn.naive_bayes.multinomialnb.html
4. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>