SnapIt - Always get the best price!

- **Team - Brute Force**
- Rachit Agrawal (IMT2020018)
- Vyom Sharma (IMT2020026)

Basic Preprocessing

- NaN values -
  - 50% of the data had NaN values for PRODUCT_BRAND - Replaced with “missing”
  - 4% of the data had NaN values for CATEGORY - Removed the rows
  - 3 rows had NaN values for PRODUCT_DESCRIPTION - Removed the rows
- No duplicate rows
- Removed 731 rows with PRODUCT_PRICE < 0

```
(main_df.isna()).sum()
```

```
PRODUCT_ID                   0
PRODUCT_NAME                 0
PRODUCT_CONDITION            0
CATEGORY                  5416
PRODUCT_BRAND           537885
SHIPPING_AVAILABILITY        0
PRODUCT_DESCRIPTION          3
PRODUCT_PRICE                0
dtype: int64
```

```
print('Removed {} rows' .format(len(main_df[main_df.PRODUCT_PRICE <=0])))
main_df = main_df[main_df.PRODUCT_PRICE > 0].reset_index(drop=True)
```

```
Removed 731 rows
```

Analysing Category Feature

- We initially split the string in category and created a list of strings separated by '/' symbol.

- We observed that only 3692 (i.e 0.3 percent) rows have more than 3 sub categories. So, we can ignore them and can consider only 3 categories for all the rows.

- We created 3 new column of subcategories as 'CAT1', 'CAT2', 'CAT3' and removed the column 'CATEGORY'.

- # Code Snippet

We first split the categories as general categories and further sub categories.

```python
main_df['CATEGORY'] = main_df['CATEGORY'].str.split("/", n = 5, expand=False)
test_df['CATEGORY'] = test_df['CATEGORY'].str.split("/", n = 5, expand=False)
```

[11]

```python
main_df['CAT1'] = main_df['CATEGORY'].str.get(0).replace('', 'missing').astype('category')
main_df['CAT2'] = main_df['CATEGORY'].str.get(1).fillna('missing').astype('category')
main_df['CAT3'] = main_df['CATEGORY'].str.get(2).fillna('missing').astype('category')
main_df.drop('CATEGORY', axis=1, inplace=True)
test_df['CAT1'] = test_df['CATEGORY'].str.get(0).replace('', 'missing').astype('category')
test_df['CAT2'] = test_df['CATEGORY'].str.get(1).fillna('missing').astype('category')
test_df['CAT3'] = test_df['CATEGORY'].str.get(2).fillna('missing').astype('category')
test_df.drop('CATEGORY', axis=1, inplace=True)
```

[14]

- # Output

CATEGORY
Beauty/Makeup/Lips
Electronics/Media/Blu-Ray
Kids/Toys/Arts & Crafts
Women/Women's Accessories/Wallets
Sports & Outdoors/Outdoors/Other

CAT1	CAT2	CAT3
Beauty	Makeup	Lips
Electronics	Media	Blu-Ray
Kids	Toys	Arts & Crafts
Women	Women's Accessories	Wallets
Sports & Outdoors	Outdoors	Other

Correlation

➔ **What**

Checked correlation between PRODUCT_PRICE and features one at a time.

➔ **Why**

Helps to select which features to keep

PRODUCT_ID

```
print(np.corrcoef(np.asarray(main_df["PRODUCT_ID"]),np.asarray(main_df["PRODUCT_PRICE"])))
```

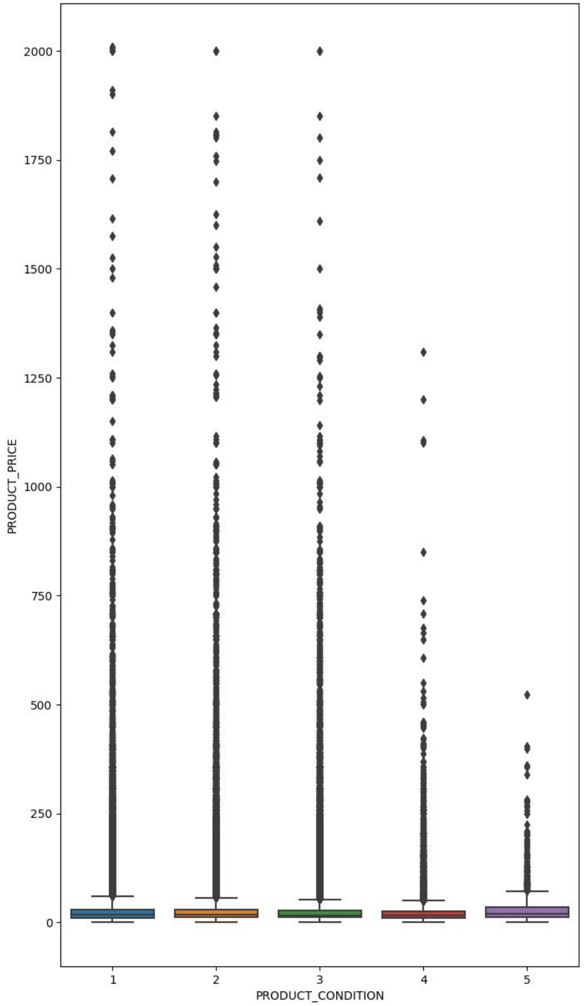
```
[[1.00000000e+00 9.46153025e-04]
 [9.46153025e-04 1.00000000e+00]]
```

The correlation between PRODUCT_ID and PRODUCT_PRICE is 0.000946 which is very less. Thus dropped PRODUCT_ID.

PRODUCT_CONDITION

```
plt.figure(figsize=(8,15))
sns.boxplot(x = main_df["PRODUCT_CONDITION"], y = main_df["PRODUCT_PRICE"])
```

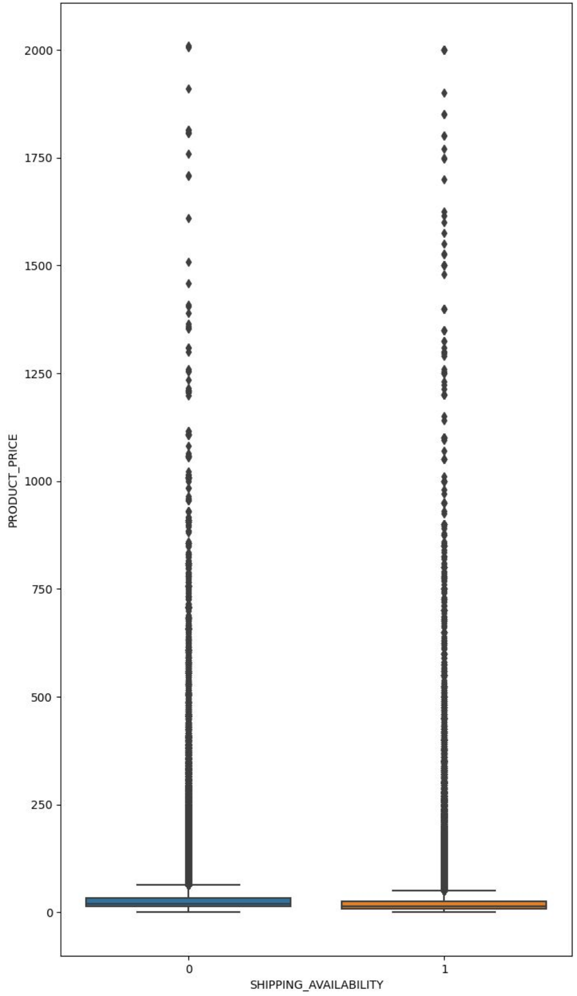
The median price decreases for product conditions 1 to 4 and slightly increases for product condition 5



SHIPPING_AVAILABILITY

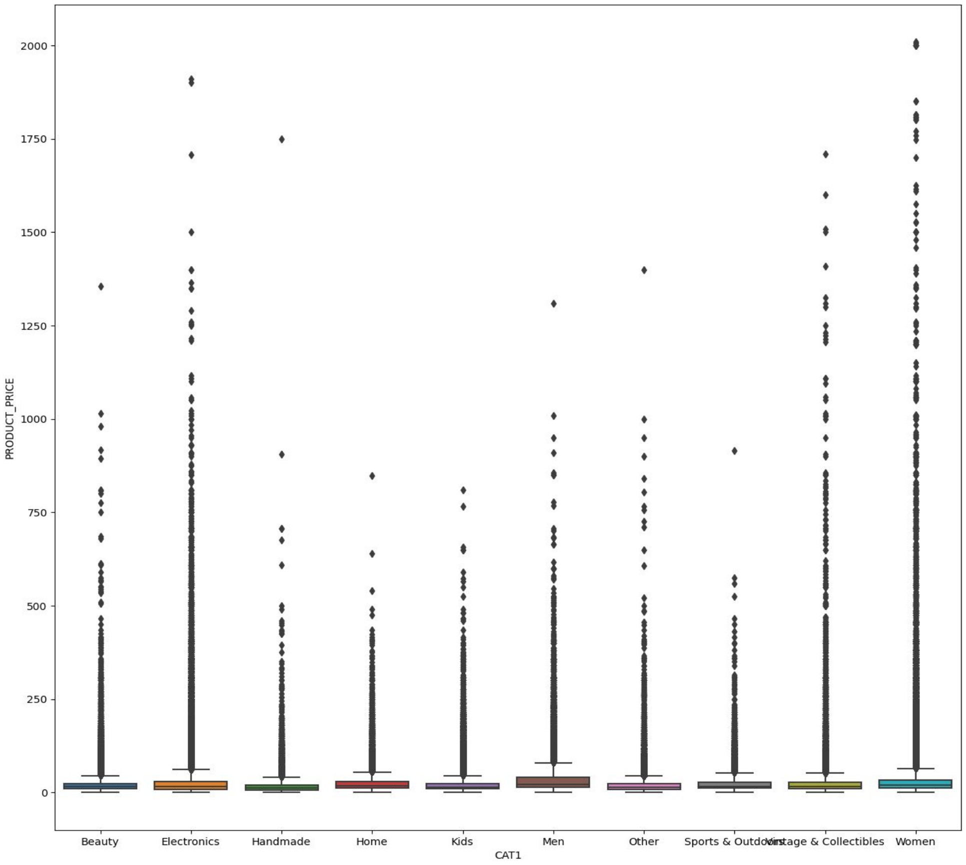
```
plt.figure(figsize=(8,15))
sns.boxplot(x = main_df["SHIPPING_AVAILABILITY"], y = main_df["PRODUCT_PRICE"])
```

The median price slightly decreases when shipping availability is available.



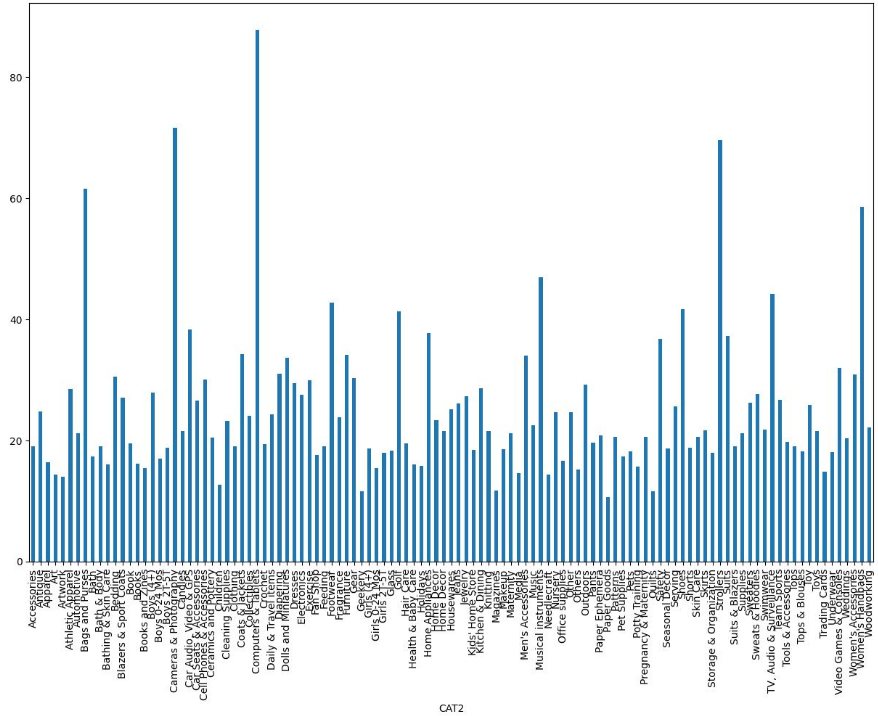
CAT1

```
plt.figure(figsize=(15,15))
sns.boxplot(x = main_df["CAT1"], y = main_df["PRODUCT_PRICE"])
```



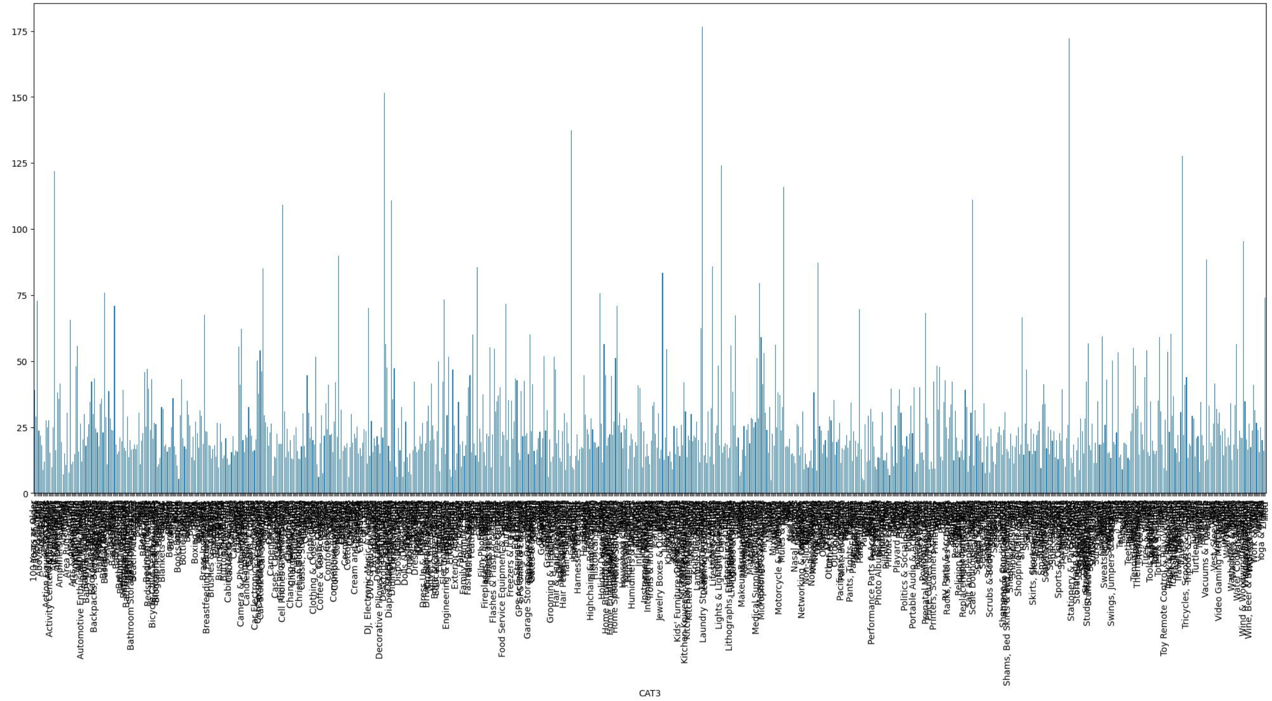
CAT2

```
plt.figure(figsize=(15,10))
cat2groupeddf = main_df.groupby("CAT2")["PRODUCT_PRICE"].mean().plot.bar()
plt.show()
```



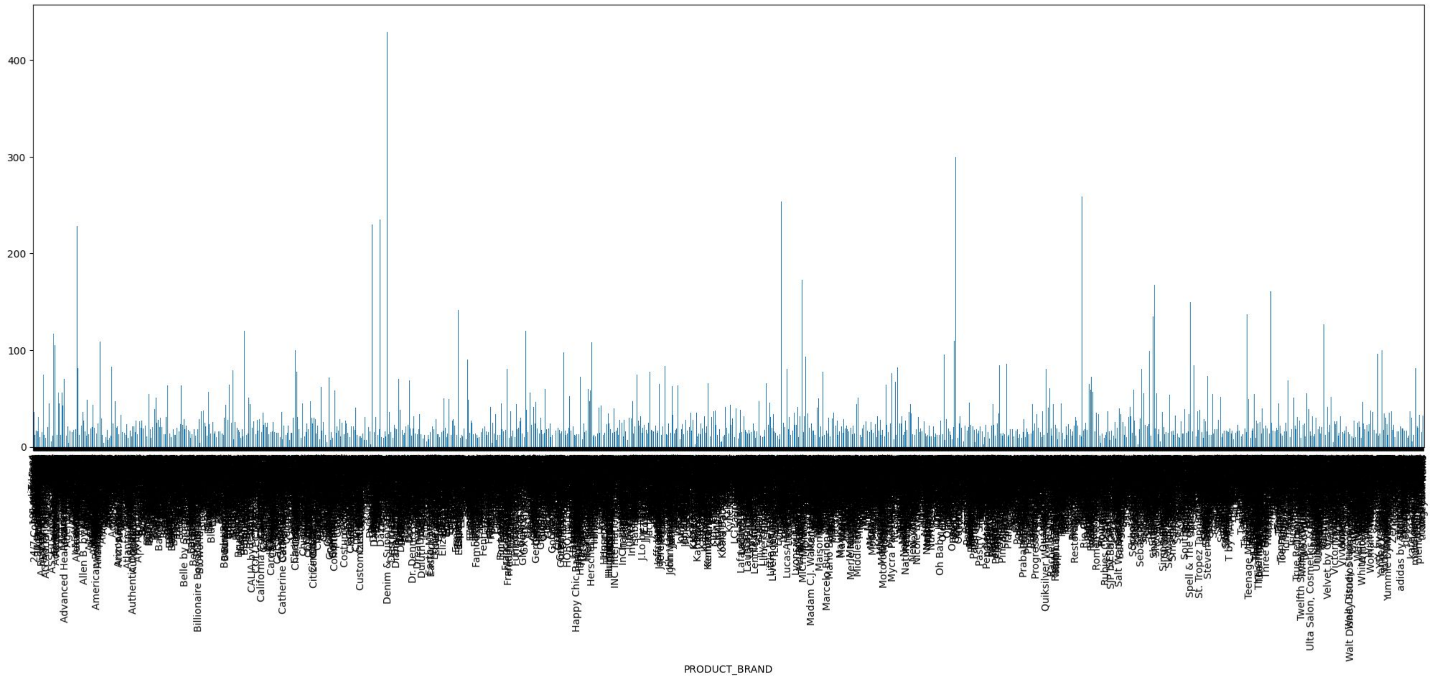
CAT3

```
plt.figure(figsize=(25,10))
cat3groupeddf = main_df.groupby("CAT3")["PRODUCT_PRICE"].mean().plot.bar()
plt.show()
```



PRODUCT_BRAND

```
plt.figure(figsize=(25,8))
brandgroupeddf = main_df.groupby("PRODUCT_BRAND")["PRODUCT_PRICE"].mean().plot.bar()
plt.show()
```



Preprocessing Text

- We preprocessed the text of 'PRODUCT_NAME' and 'PRODUCT_DESCRIPTION'. We removed non-alphanumeric characters, regular expressions, stopwords, tab space, newline from the text.

```python
def PreprocessName(name_col):
    preprocessed_names = []
    for sentence in tqdm(name_col.values):
        s = sentence.replace('\\r', ' ')
        s = s.replace('\\"', ' ')
        s = s.replace('\\n', ' ')
        s = re.sub('[^A-Za-z0-9]+', ' ', s)
        preprocessed_names.append(s.lower().strip())
    return preprocessed_names
```

[26] Python

```python
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

stopwords = stopwords.words('english')
def PreprocessDescription(desc_col):
    preprocessed_descs = []
    for sentence in tqdm(desc_col.values):
        s = sentence.replace('\\r', ' ')
        s = s.replace('\\"', ' ')
        s = s.replace('\\n', ' ')
        s = re.sub('[^A-Za-z0-9]+', ' ', s)
        s = ' '.join(e for e in s.split() if e not in stopwords)
        preprocessed_descs.append(s.lower().strip())
    return preprocessed_descs
```

[27] Python

- **We preprocessed the text of CATEGORY. We removed blank spaces from the text.**

```python
def PreprocessCategory(cat_col):
    catogories = list(cat_col)

    cat_list = []
    for i in tqdm(catogories):
        i = re.sub('[^A-Za-z0-9]+', ' ', i)
        i = i.replace(' ','')
        cat_list.append(i.strip())

    return cat_list
```

[28] Python

- **Result : We observed that after text preprocessing the distinct values in 'PRODUCT_NAME', 'PRODUCT_DESCRIPTION' and 'CAT3' decreased.**

```python
[25] for col in main_df:
         print(col, len(main_df[col].unique()))
```

```
PRODUCT_NAME 1048068
PRODUCT_CONDITION 5
PRODUCT_BRAND 4616
SHIPPING_AVAILABILITY 2
PRODUCT_DESCRIPTION 1089018
PRODUCT_PRICE 798
CAT1 10
CAT2 113
CAT3 863
```

```python
[33] for col in main_df:
         print(col, len(main_df[col].unique()))
```

```
PRODUCT_NAME 956790
PRODUCT_CONDITION 5
PRODUCT_BRAND 4613
SHIPPING_AVAILABILITY 2
PRODUCT_DESCRIPTION 1063367
PRODUCT_PRICE 797
CAT1 10
CAT2 113
CAT3 861
```

One Hot Encoding

- Finally, we one hot encoded the columns ‘PRODUCT_BRAND’, ‘CAT1’, ‘CAT2’ and ‘CAT3’.

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(lowercase=False, binary=True)
train_brand_ohe = vectorizer.fit_transform(main_df['PRODUCT_BRAND'].values)
test_brand_ohe = vectorizer.transform(test_df['PRODUCT_BRAND'].values)
```

```
train_cat1_ohe = vectorizer.fit_transform(main_df['CAT1'].values)
test_cat1_ohe = vectorizer.transform(test_df['CAT1'].values)
```

```
train_cat2_ohe = vectorizer.fit_transform(main_df['CAT2'].values)
test_cat2_ohe = vectorizer.transform(test_df['CAT2'].values)
```

```
train_cat3_ohe = vectorizer.fit_transform(main_df['CAT3'].values)
test_cat3_ohe = vectorizer.transform(test_df['CAT3'].values)
```

PRODUCT_NAME and PRODUCT_DESCRIPTION

```python
from sklearn.feature_extraction.text import TfidfVectorizer

# Creating a TF-IDF vectorizer with 1,2, and 3 ngrams together for PRODUCT_NAME
vectorizer = TfidfVectorizer(ngram_range=(1, 3), min_df=3, max_features=250000)

tfidf_name = vectorizer.fit_transform(main_df['PRODUCT_NAME'].values)
test_tfidf_name = vectorizer.transform(test_df['PRODUCT_NAME'].values)

# Creating a TF-IDF vectorizer with 1,2, and 3 ngrams together for PRODUCT_DESCRIPTION
vectorizer = TfidfVectorizer(ngram_range=(1, 3), min_df=5, max_features=500000)

tfidf_description = vectorizer.fit_transform(main_df['PRODUCT_DESCRIPTION'].values)
test_tfidf_description = vectorizer.transform(test_df['PRODUCT_DESCRIPTION'].values)
```

- **Created sparse matrix containing TF-IDF scores for 1,2 and 3 grams for each document**

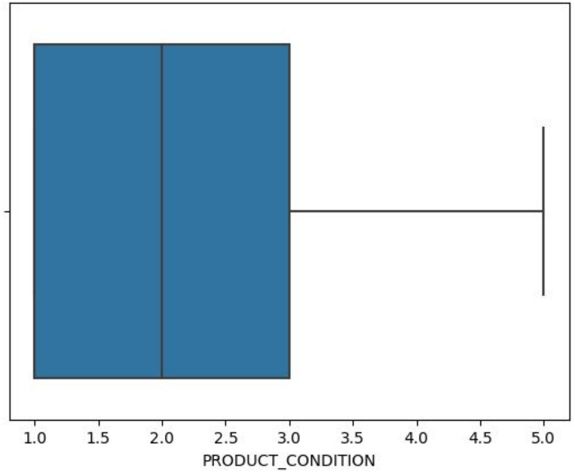
$$\mathrm{idf}(t) = \log \frac{n}{\mathrm{df}(t)} + 1 \qquad w_{i,j} = tf_{i,j} \times idf_i$$

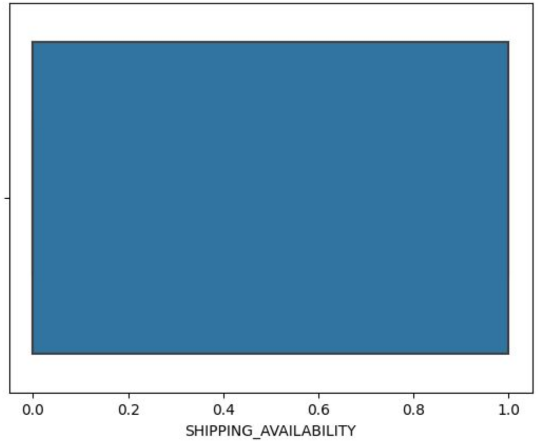
Outliers, Normalisation and Standardisation

- Only PRODUCT_CONDITION and SHIPPING_AVAILABILITY needs to be checked since other features are one hot encoded and PRODUCT_NAME and PRODUCT_DESCRIPTION have been converted to TF-IDF vectors.

- No Normalisation or Standardisation required

```
cols = ['PRODUCT_NAME','PRODUCT_DESCRIPTION','PRODUCT_PRICE','PRODUCT_BRAND','CAT1','CAT2','CAT3']
for col in main_df:
    if col in cols:
        continue
    sns.boxplot(x=main_df[col])
    plt.show()
```





References

- https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0
- https://www.analyticsvidhya.com/blog/2021/05/feature-engineering-how-to-detect-and-remove-outliers-with-python-code/
- https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
- https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html