A Project Report

On

Price Estimation Model for Used Household Items

BY

Ritvik Raj Padige

17XJ1A0542

Rakshith S.

17XJ1A0539

Pruthvik Shashank

17XJ1A0536

Under the supervision of

Dr N. Raghu Kishore

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF

PR 401 / PR 402: PROJECT TYPE COURSE

LEADER ■ ENTREPRENEUR ■ INNOVATOR

ECOLE CENTRALE COLLEGE OF ENGINEERING

MAHINDRA UNIVERSITY

HYDERABAD

(May 2021)

ACKNOWLEDGMENTS

In performing our assignment, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much pleasure. We would like to show our gratitude to our professor **Dr N. Raghu Kishore, Dept. Of Computer Science & Engineering, Mahindra University** for giving us a good guideline for the paper throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment. In addition, a thank you to Professor **Dr N. Raghu Kishore**, who introduced us to the Methodology of work, and whose passion for the "underlying structures" had a lasting effect. We also thank the institution for consent to include copyrighted pictures as a part of our paper. Many people, especially our classmates and team members themselves, have made valuable comments and suggestions on this proposal which inspired us to improve our paper. We thank all the people for their help directly and indirectly to complete our paper.

Ecole Centrale College of Engineering

Mahindra University

Hyderabad

Certificate

This is to certify that the project report entitled "**Price Estimation Model for Used Household Items**" submitted by **Mr Ritvik Raj Padige (HT No. 17XJ1A0542), Mr Rakshith S. (HT No. 17XJ1A0539), Mr Pruthvik Shashank (HT No. 17XJ1A0536)** in partial fulfilment of the requirements of the course PR 401/PR 402, Project Course, embodies the work done by him/her under my supervision and guidance.

(Dr. N. Raghu Kishore)

Ecole Centrale School of Engineering, Hyderabad

Date:

ABSTRACT

Prediction of future movement of prices has been a subject matter of many research works. Propositions are illustrating that, if appropriately modelled, prices can be predicted with a high level of accuracy. There is also a gamut of literature on technical analysis of prices where the objective is to identify patterns in price movements and profit from them. In this work, we propose a hybrid approach for price prediction using machine learning-based methods.

Pricing is the biggest headache when it comes to selling products online. It is where people feel the most pressure to get the right price and least certain that they are doing a good job. The pressure is intensified because, for the most part, people believe that they do not have control over price. This is where we found the motivation for the project to come up with a price prediction model which gives you accurate pricing for a product based on its various parameters using basic principles of machine learning tools like sci-kit-learn which features various classification, regression and clustering algorithms and data analysis tools like pandas, seaborn, matplotlib and NumPy.

E-Commerce sites implement recommendation structures for recommending products to the buyer and enhance their sales. This may be efficaciously carried out if the site has standardized its pricing scheme and suggests this price to the seller. The report aims to predict the prices of products with the various characteristics of the product.

The following are covered by the report:

- Machine learning in the domain of E-Commerce
- Natural Language Processing in E-Commerce
- Data and Data Handling
- Prediction Algorithms
- MSE and R^2

Contents

1.0 INTRODUCTION

E-commerce is enjoying a crucial role these days. Why should one walk to a retail store once we can purchase things online while sitting at home? This type of thinking has made E-commerce sites popular. E-commerce websites make use of promotional activities like a flash sale, etc. to attract more and more customers. When a customer visits their site for buying some products, they recommend some other product along with it which will be compatible with the product being purchased. This makes the customer aware of the other products available on the e-commerce website and may also tempt the person to buy the product even if it was not intended. In this way, sales of the e-commerce companies are increased, this is the main reason why recommendation systems play an important role for an e-commerce website.

Also, companies like Amazon, Alibaba which sell brand new products have standard pricing across all the products, so customers show interest in buying the recommended product as the pricing all across the other websites and retail stores is also the same. This is not the case with e-commerce websites selling refurbished products. On websites like OLX, eBay, etc. the resale products do not have standardized pricing across all products as the products being sold on their websites are not coming directly from a manufacturer but instead, are coming from a person who has already used it. When a person is selling a product, he/she will always quote the price of the product on what he/she believes to be appropriate. Due to this some of the products being sold on such websites are overpriced as the seller does not consider all the qualities of the product. Looking from a buyer's point of view, why will he/she purchase an overpriced product.

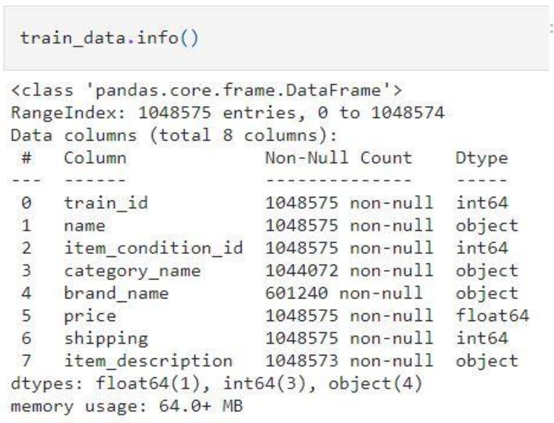
For example, suppose an Apple iPhone 12 is being sold by the manufacturer at Rs 75,000, while six months used iPhone is being sold by a seller on OLX at Rs 70,000. Now in this case the customer will always prefer a brand new iPhone as it is only 5,000 more than the resale iPhone. This creates a loss to the e-commerce websites selling resale products as they are not able to get their share of profit until the products on their website are being sold. As these websites are not having standardized pricing, they are not able to recommend products to the customers. To pass these problems of the e-commerce websites selling resale products this paper implements a market basket-based recommendation model which will recommend those standardized price products and price prediction algorithm for standardizing the pricing of the resale products. Coming to the prediction rule this paper implements 3 models specifically ridge regression, light gradient boosting machine (LGBM) and XGBM. The data used for predicting the worth is from the website of Kaggle which is availed by Mercari one of the leading e-commerce websites in Japan.

2.0 Data and Data Handling

The data consists of 1048575 rows. After using exploratory data analysis, it was concluded that shipping barely plays any role in predicting the worth. So, the name of the merchandise, brand name, classes and item description were the key parameters for predicting the price. Name of the merchandise, brand name, classes and item description was matter features, tongue process techniques like count vectorizer, TF-IDF vectorizer, text to sequences, etc were accustomed to convert those textual data into a numerical format. Since value could be a continuous variable the metrics used for analysis is Mean Square Error(MSE) and R squared.

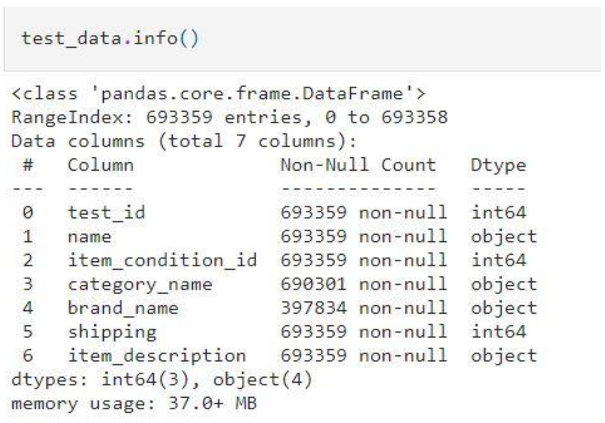
2.0.1 Dataset:

We picked a dataset from Kaggle and we will use some basic data handling techniques to understand our problem better and get a better solution according to the data.



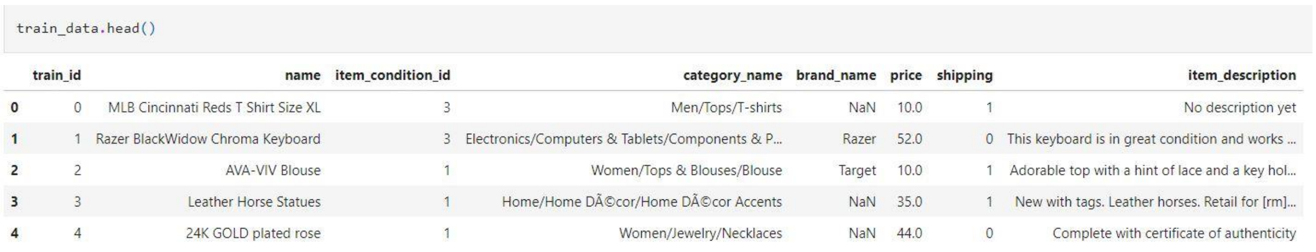
```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column             Non-Null Count    Dtype  
---  ------             --------------    -----  
 0   train_id           1048575 non-null  int64  
 1   name               1048575 non-null  object 
 2   item_condition_id  1048575 non-null  int64  
 3   category_name      1044072 non-null  object 
 4   brand_name         601240 non-null   object 
 5   price              1048575 non-null  float64
 6   shipping           1048575 non-null  int64  
 7   item_description   1048573 non-null  object 
dtypes: float64(1), int64(3), object(4)
memory usage: 64.0+ MB
```



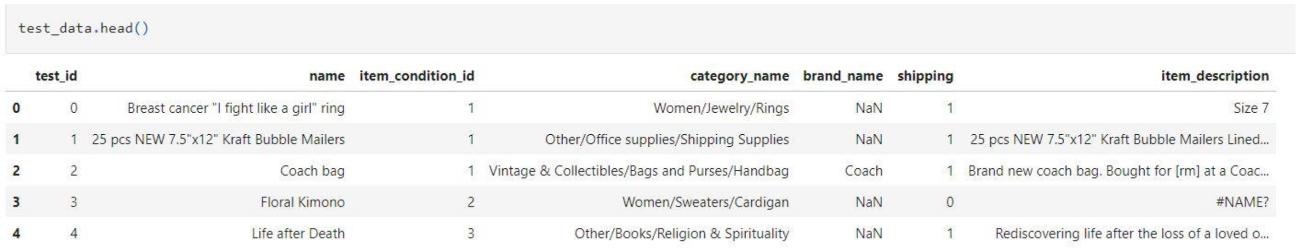
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 693359 entries, 0 to 693358
Data columns (total 7 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   test_id            693359 non-null  int64 
 1   name               693359 non-null  object
 2   item_condition_id  693359 non-null  int64 
 3   category_name      690301 non-null  object
 4   brand_name         397834 non-null  object
 5   shipping           693359 non-null  int64 
 6   item_description   693359 non-null  object
dtypes: int64(3), object(4)
memory usage: 37.0+ MB
```



```
train_data.head()
```

	train_id	name	item_condition_id	category_name	brand_name	price	shipping	item_description
0	0	MLB Cincinnati Reds T Shirt Size XL	3	Men/Tops/T-shirts	NaN	10.0	1	No description yet
1	1	Razer BlackWidow Chroma Keyboard	3	Electronics/Computers & Tablets/Components & P...	Razer	52.0	0	This keyboard is in great condition and works ...
2	2	AVA-VIV Blouse	1	Women/Tops & Blouses/Blouse	Target	10.0	1	Adorable top with a hint of lace and a key hol...
3	3	Leather Horse Statues	1	Home/Home DÃ©cor/Home DÃ©cor Accents	NaN	35.0	1	New with tags. Leather horses. Retail for [rm]...
4	4	24K GOLD plated rose	1	Women/Jewelry/Necklaces	NaN	44.0	0	Complete with certificate of authenticity

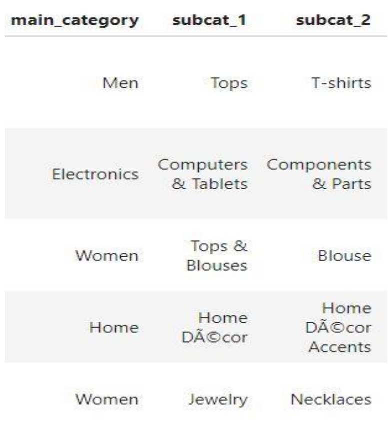


```
test_data.head()
```

	test_id	name	item_condition_id	category_name	brand_name	shipping	item_description
0	0	Breast cancer "I fight like a girl" ring	1	Women/Jewelry/Rings	NaN	1	Size 7
1	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers	1	Other/Office supplies/Shipping Supplies	NaN	1	25 pcs NEW 7.5"x12" Kraft Bubble Mailers Lined...
2	2	Coach bag	1	Vintage & Collectibles/Bags and Purses/Handbag	Coach	1	Brand new coach bag. Bought for [rm] at a Coac...
3	3	Floral Kimono	2	Women/Sweaters/Cardigan	NaN	0	#NAME?
4	4	Life after Death	3	Other/Books/Religion & Spirituality	NaN	1	Rediscovering life after the loss of a loved o...

2.0.2 Data Pre-Processing :

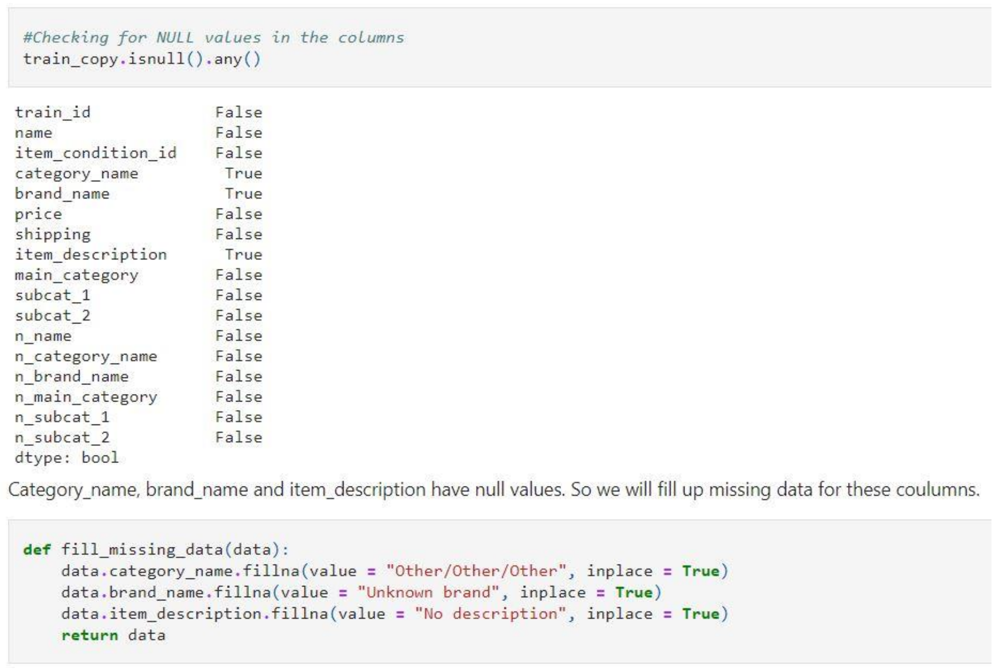
In Data Pre-Processing we will split the category name into three parts: (1) Main Category, (2) First Sub-Category, (3) Second Sub-Category. Whenever blanks are found, they will be replaced as "No Label" for these three. Then we will apply label encoding to them.



main_category	subcat_1	subcat_2
Men	Tops	T-shirts
Electronics	Computers & Tablets	Components & Parts
Women	Tops & Blouses	Blouse
Home	Home DÃ©cor	Home DÃ©cor Accents
Women	Jewelry	Necklaces

2.0.3 Data Cleaning:

We will apply basic data cleaning like Filling up missing data, dropping NA etc.



```
#Checking for NULL values in the columns
train_copy.isnull().any()
```

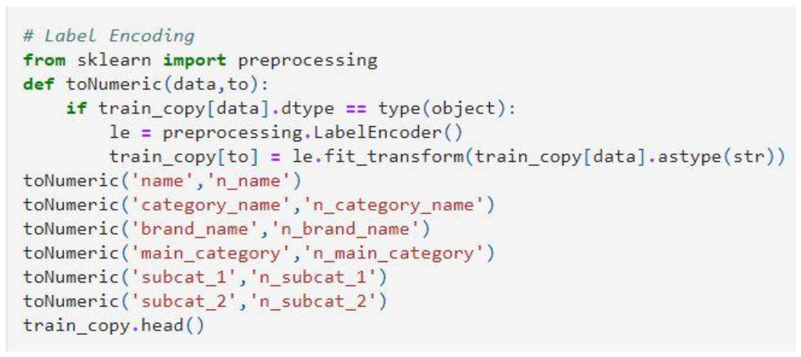
```
train_id              False
name                  False
item_condition_id     False
category_name          True
brand_name             True
price                 False
shipping              False
item_description       True
main_category         False
subcat_1              False
subcat_2              False
n_name                False
n_category_name       False
n_brand_name          False
n_main_category       False
n_subcat_1            False
n_subcat_2            False
dtype: bool
```

Category_name, brand_name and item_description have null values. So we will fill up missing data for these coulumns.

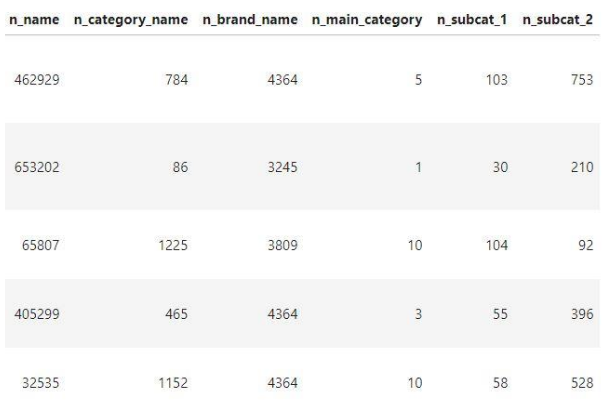
```
def fill_missing_data(data):
    data.category_name.fillna(value = "Other/Other/Other", inplace = True)
    data.brand_name.fillna(value = "Unknown brand", inplace = True)
    data.item_description.fillna(value = "No description", inplace = True)
    return data
```

2.0.4 Label Encoding:

Label Encoding refers to converting the labels into a numeric form to convert it into a machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.



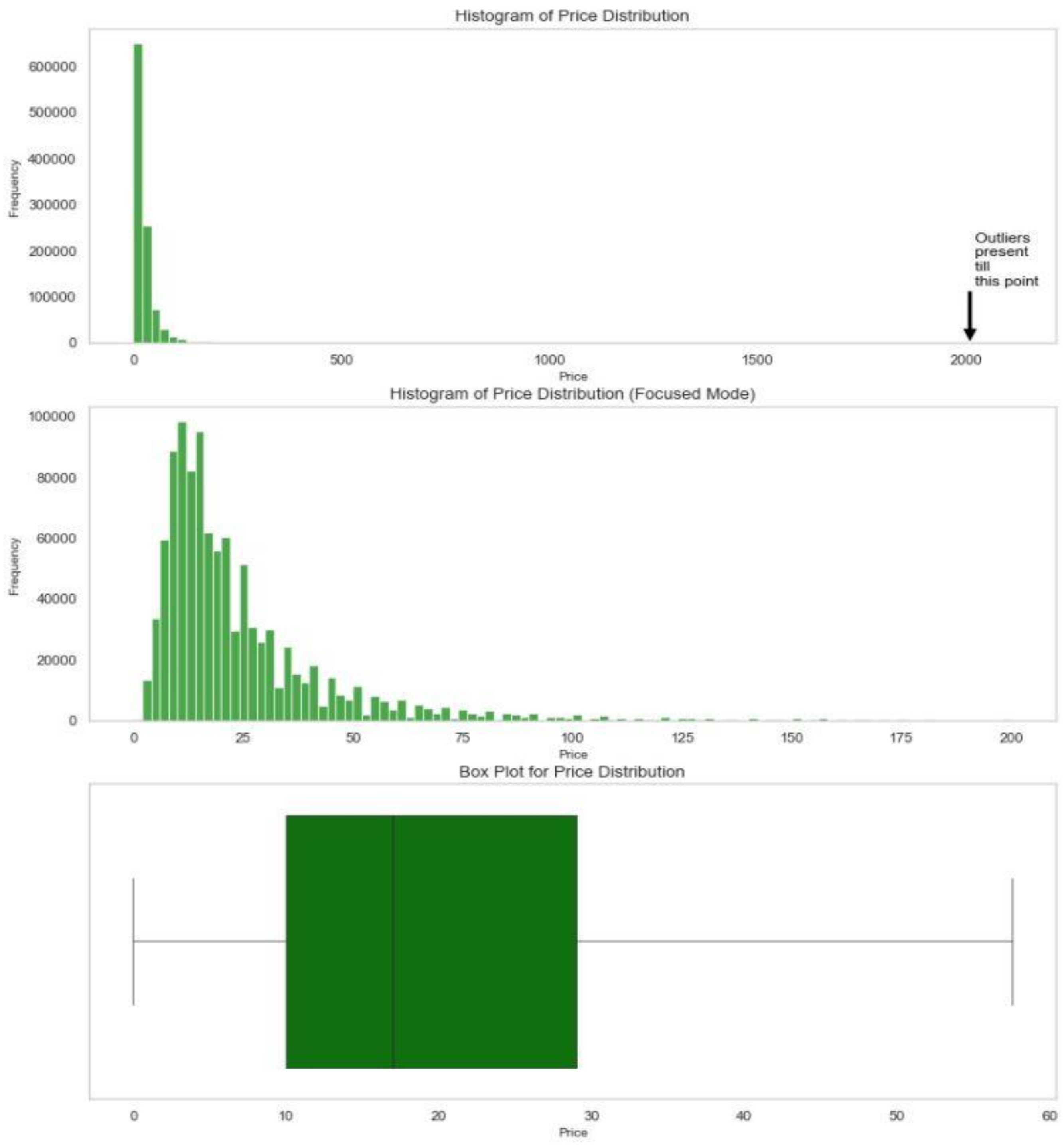
```
# Label Encoding
from sklearn import preprocessing
def toNumeric(data,to):
    if train_copy[data].dtype == type(object):
        le = preprocessing.LabelEncoder()
        train_copy[to] = le.fit_transform(train_copy[data].astype(str))
toNumeric('name','n_name')
toNumeric('category_name','n_category_name')
toNumeric('brand_name','n_brand_name')
toNumeric('main_category','n_main_category')
toNumeric('subcat_1','n_subcat_1')
toNumeric('subcat_2','n_subcat_2')
train_copy.head()
```



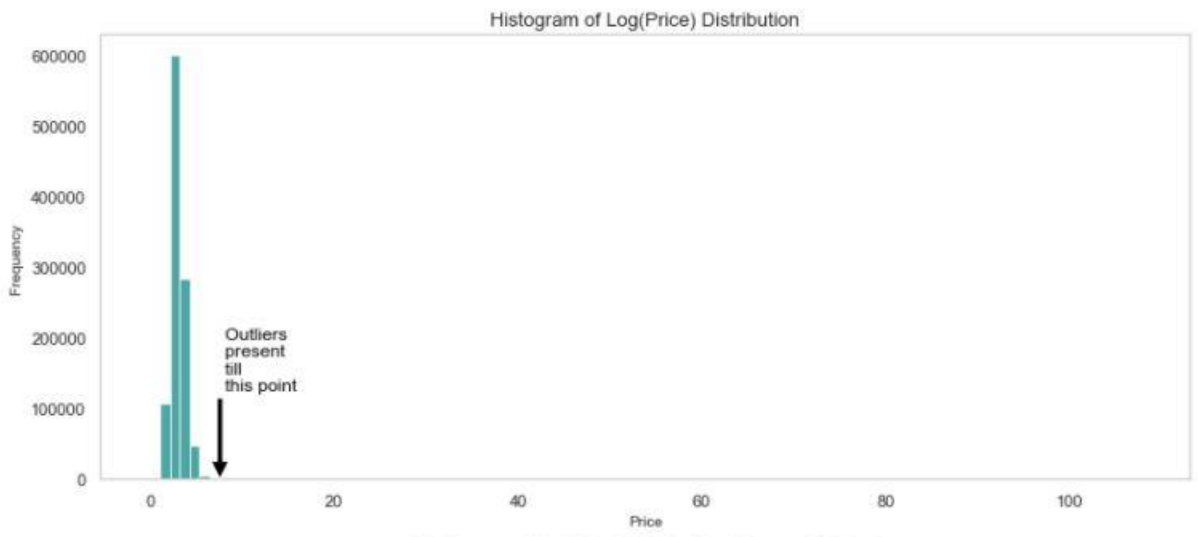
n_name	n_category_name	n_brand_name	n_main_category	n_subcat_1	n_subcat_2
462929	784	4364	5	103	753
653202	86	3245	1	30	210
65807	1225	3809	10	104	92
405299	465	4364	3	55	396
32535	1152	4364	10	58	528

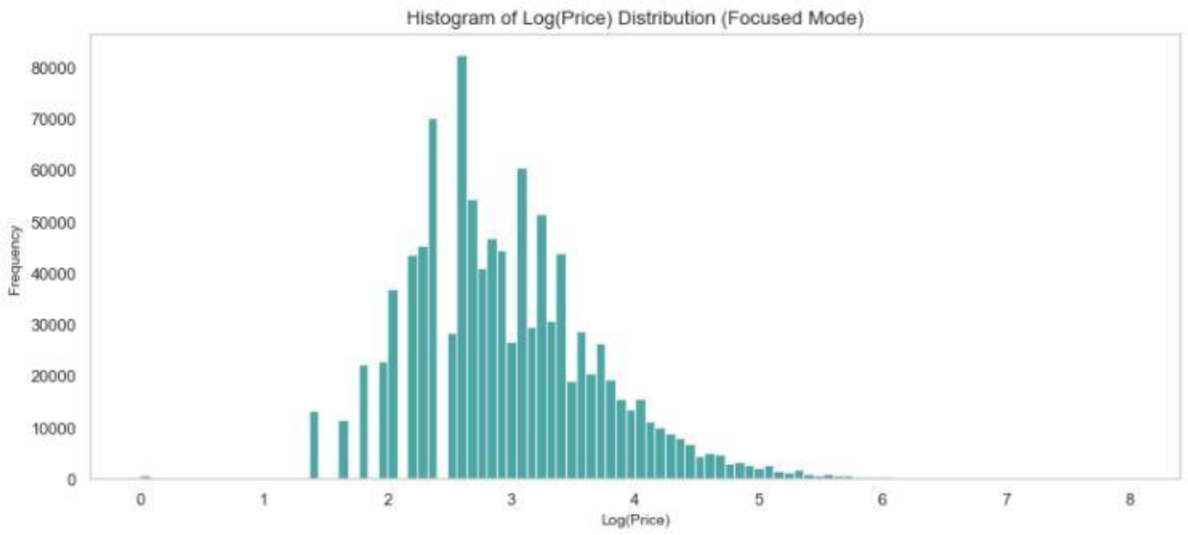
2.0.5 Exploratory Data Analysis :

We will proceed with doing some EDA now to explore some interesting findings.

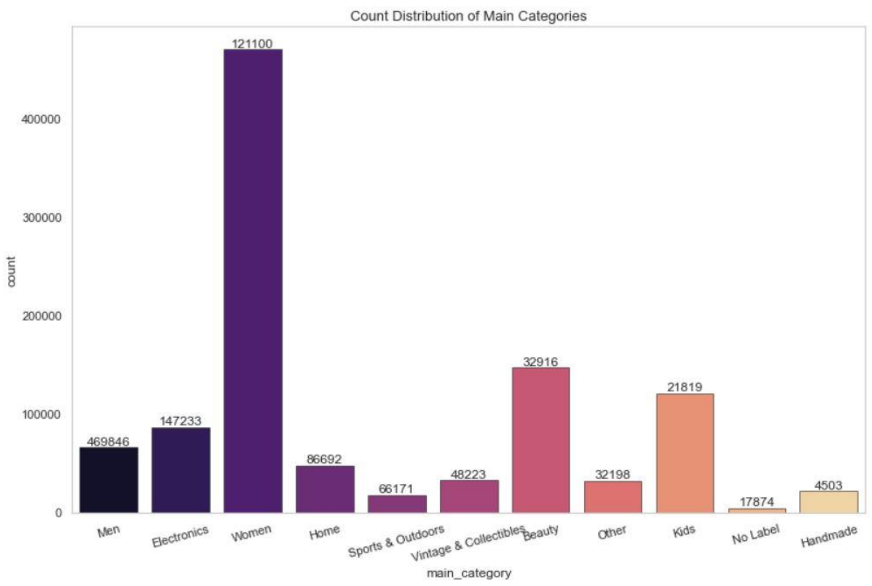


After analysing the price distribution histogram, we can infer that the distribution seems to be skewed and for accurate results, the histogram needs to be uniform for that we will be applying the log function.

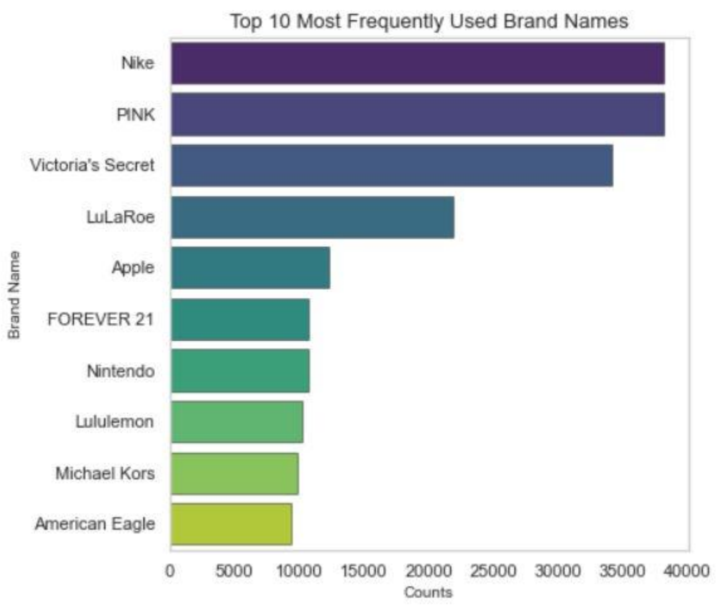


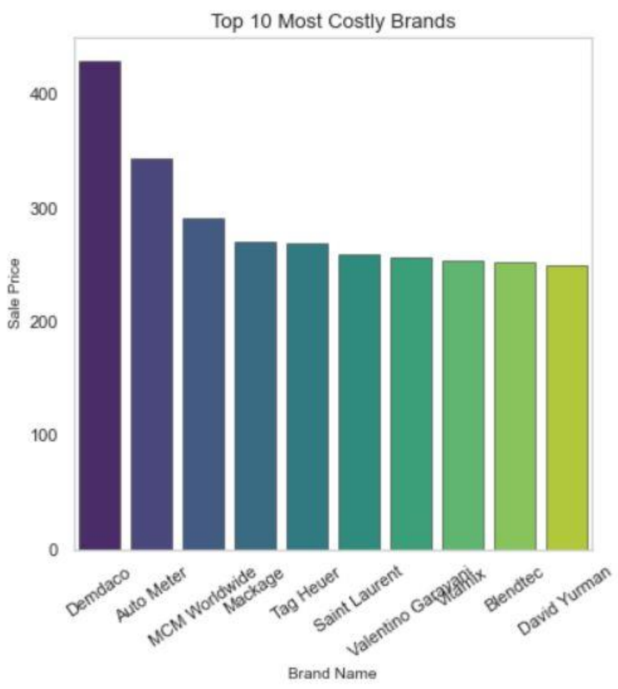


Now the transformed price distribution has taken the symmetric shape and can be said that it is following Normal Distribution. We can work with this.

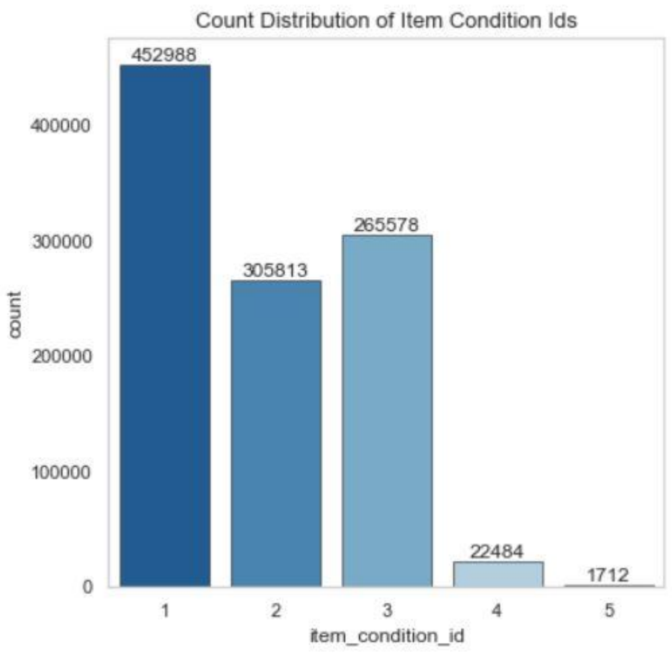


We can see that the majority of the items comprise of the Women, Beauty and Kids categories





"Nike" is the most widely used brand whereas "Demdaco" is the costliest brand in the lot.



"1" is the most widely used item_condition_id. Products having item_condition_ids "1" and "5" are costly.

2.1 Machine Learning in the Domain of E-Commerce

We may or may not have acknowledged that Machine Learning already influences you daily, nearly everywhere. We have seen it daily out on Amazon with its' customized product suggestions. And Facebook with its facial recognition program for tagging human beings in photos. Which is made feasible through the utility of an idea known as the Internet of Things. Implementing Machine Learning strategies to E-Commerce is an enormous new possibility.

As pricing in e-commerce is very dynamic, creator **Bauer and Jannach** (2018) has applied a version to optimize the pricing of sparse and noisy information of E-Commerce. For enforcing this the writer has made use of Parameters, Price adjustments through the years for a specific product and associated products. Bootstrap-based Confidence is blended with Confidence Bayesian Inference and Kernel Regression. The model proposed by the author was successful, the profit increased by 28.04% of revenue in just 4 months.

On the other hand, using recommendation and pricing as the competitiveness parameter author **Jiang et al.** (2015) has redesigned promotion strategy for e-commerce. Three models are used by the author Online Promotion and Recommendation (OPR), Promotion with No Recommendation (PNR) and Minimum cost Ratios (MCR). Out of these OPR model outperformed the other two models by 27.7% and 10.3%. According to the author, customers ought to be inspired to shop for merchandise by giving some discount on them and at an equivalent time, alternative non-discounted merchandise ought to be counselled to the shoppers. By doing this the loss that is occurred by giving discount are often recovered by recommending alternative non-discounted merchandise. The parameters utilized by the author for implementing this area unit current value, product value and reservation value.

2.2 Natural Language Processing in E-Commerce

Today's E-Commerce biggies can index a huge range of product names or descriptions to resource product search; however, they do not continually give satisfactory results. This traditional search method wants continuous improvement through the addition of discourse awareness to provide correct results. NLP will be accustomed to add this 'awareness' to understand both the searches and the products in a typical application.

Author Tripathy et al. (2015) has enforced text classification exploitation machine learning classification algorithms specifically naive Bayes and support vector machine. These algorithms were accustomed classify the sentiments into positive and negative. For processing the text, the author first converted the text into tokens, removed the stop words and therefore the punctuation marks then used count vectorizer and TF-IDF vectorizer for changing the textual information into numerical type. The results showed that the support vector machine (94%) outperformed naive Bayes (89.5%) accuracy for classifying the sentiments. For classifying text three different feature extraction techniques were employed by author Dzisevi and Eok (2019) which are term frequency-inverse document frequency (TF-IDF), linear discriminant analysis (LDA) and latent linguistics analysis (LSA) over a neural network classifier.

For implementing the model ten thousand rows advertisement information was acquired from Lithuanian advertisement website for classifying into twenty classes. TF-IDF vectorizer outperformed the other 2 feature extraction techniques by serving the model to realize a ninety-one-accuracy score.

A few examples are discussed below:

- If a user phrased his search query as “black track-suit”, the search engine may not essentially deliver results about tracksuit but may show results about tracks and suits, because the concept of a tracksuit is something the engine might not know unless NLP based contextual awareness is integrated.

- For a search like “candle stand” most current eCommerce search engines will deliver results for candles too. NLP can help the search engine understand that the word ‘candle’ is describing the word ‘stand’.

A search query for ‘gluten-free bread’ would involve understanding the context of the words, and telling the engine what you do not want, this type of negation in search is very hard for traditional search engines without NLP.

3.0 Prediction Algorithms

3.0.1 Ridge Regression

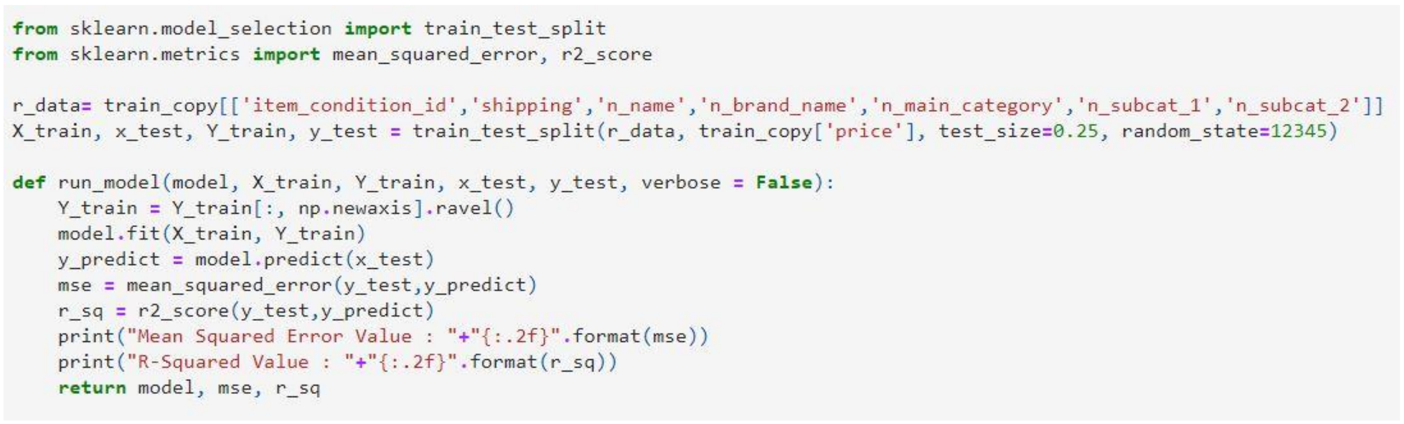
For big data to understand the computational speed of the algorithm Author Chiang et al. (2018) has implemented ridge regression. The data used for implementing the model is from the bureau of transportation statistics and the federal aviation administration which were pretty much large and consisted of the data of each flight from the year 2010 to 2015. Ridge regression is used for predicting the arrival and departure delay of the airlines. The ridge regression method proposed by the author was successful in predicting the target variables with a mean squared error of 168,632.10 and a mean absolute error of 394,368.89. The author concludes that ridge regression requires very little memory, has faster computing speed and provides accurate results. Ridge Regression is similar to Linear Regression except that we introduce a small amount of bias. In return for said bias, we get a considerable drop in the variance. In other words, by starting with a slightly worse fit, Ridge Regression performs better against data that does not exactly follow the same pattern as the data the model was trained on.

Adding bias is often referred to as regularization. As the name implies, regularization is used to develop a model that excels at predicting targets for data that follows a regular pattern rather than specific. Said another way, the purpose of regularization is to prevent overfitting. Overfitting tends to occur when we use a higher degree polynomial than what is needed to model the data.

3.0.2 LGBM Regression

LGBM is a gradient boosting based framework which used algorithms based on tree learning. LGBM grows vertically while other tree-based algorithms grow horizontally LGBM is implemented in this paper for the following reasons higher efficiency and training speed is fast, memory usage is low, great accuracy, able to handle huge data set Advantages of histogram-based algorithms include the following:

- Reduced cost of calculating the gain for each split.
- Pre-sort-based algorithms have time complexity O(#data)
- Computing the histogram has time complexity O(#data), but this involves only a fast sum-up operation. Once the histogram is constructed, a histogram-based algorithm has time complexity O(#bins), and #bins are far smaller than #data.
- Use histogram subtraction for further speedup.
- To get one leaf's histograms in a binary tree, use the histogram subtraction of its parent and its neighbour.
- So, it needs to construct histograms for only one leaf (with smaller #data than its neighbour). It then can get histograms of its neighbour by histogram subtraction with small cost (O(#bins))
- Reduce memory usage.
- Replaces continuous values with discrete bins. If #bins are small, can use small data type, e.g., uint8_t, to store training data.
- No need to store additional information for pre-sorting feature values.
- Reduce communication cost for parallel learning.



```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

r_data= train_copy[['item_condition_id','shipping','n_name','n_brand_name','n_main_category','n_subcat_1','n_subcat_2']]
X_train, x_test, Y_train, y_test = train_test_split(r_data, train_copy['price'], test_size=0.25, random_state=12345)

def run_model(model, X_train, Y_train, x_test, y_test, verbose = False):
    Y_train = Y_train[:, np.newaxis].ravel()
    model.fit(X_train, Y_train)
    y_predict = model.predict(x_test)
    mse = mean_squared_error(y_test,y_predict)
    r_sq = r2_score(y_test,y_predict)
    print("Mean Squared Error Value : "+"{:.2f}".format(mse))
    print("R-Squared Value : "+"{:.2f}".format(r_sq))
    return model, mse, r_sq
```

3.0.3 XGB Regression

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular information. XGBoost is an implementation of gradient boosted call trees designed for speed and performance. XGBoost stands for eXtreme Gradient Boosting. The implementation of the formula was built for the potency of computing time and memory resources. A design goal was to create the most effective use of accessible resources to coach the model.

Some key formula implementation options include:

• Sparse Aware implementation with automatic handling of missing information values.

• Block Structure to support the parallelization of tree construction.

• Continued coaching so that you'll be able to further boost an already fitted model on new information.

XGBoost dominates structured or tabular datasets on classification and regression predictive modelling issues. Thus, we tend to are predicting we are going to get the most effective results from this prediction formula.

3.1 Proposed Model

We propose a model using Advanced Text-Pre-Processing in which with the combined set of train and test data we apply the count vectorizer and it will help us get the list of all possible words. Text pre-processing processes are as follows.

- Removing punctuations
- Removing digits
- Removing stop words
- Changing to lower case words
- Lemmatization or stemming

This combined with Tfidfvectorizer/ LabelBinarizer.

- Count Vectorizer counts word frequencies.
- TF-IDF Vectorizer gives more significance (puts more weights) on rare words, and less significance (puts lesser weights) on frequent words.
- Label Binarizer converts labels into numeric representations e.g. "A, B, C" -> [1,2,3] we will get a modified model which will I believe give the results which are much more accurate.

4.0 Stepwise Process

Step 1

Importing training and testing datasets using Pandas.

Step 2

Converting Category name into 3 subcategories as category name has at most 3 categories separated by a slash.

Step 3

Exploratory data analysis using matplotlib and seaborn packages.

Step 4

Label Encoder used to convert all non-numerical labels to numerical labels in different named columns like n_name, n_category_name etc. this is done using sklearn package using preprocessing and .LabelEncoder() module

Step 5

Data Cleaning using fillna function for filling nulling values and check for null values. In our case, we have nulls in Category_name, brand_name and item_description. So, we will fill up the missing data for these columns.

Step 6

Log transformation of the price using np.log1p for getting more accurate results as skewed distribution gets converted to a normal uniform distribution.

Step 7

Finding the correlation between different columns using corr() function and plotting them on an sns seaborn heatmap. This function automatically ignores any non-numerical values.

Step 8

Define and train model, define MSE and R_2 score functions for predictive analysis.

Step 9

Advanced text pre-processing and Applying CountVectorizer, Tfidfvectorizer, LabelBinarizer on a data frame.

Step 10

Define the model and complete training then check results for all 3 models again to compare results.

4.0 MSE and R Squared used for predictive analysis.

The **Mean Squared Error (MSE)** or **Mean Squared Deviation (MSD)** of an estimator measures the average of error squares i.e., the average squared difference between the estimated values and true value. It is a risk function, corresponding to the expected value of the squared error loss. It is always non – negative and values close to zero are better. RMSE is simply the square root of MSE and shows us the efficiency of the model.

$$\text{MSE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} (\underbrace{y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i}_{\text{actual value}})^2$$

the **coefficient of determination** denoted *R^2* or *r^2* and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

0% indicates that the model explains none of the variability of the response data around its mean.

100% indicates that the model explains all the variability of the response data around its mean.

$$R^2 = 1 - \frac{\overset{\text{Sum Squared Regression Error}}{SS_{Regression}}}{\underset{\text{Sum Squared Total Error}}{SS_{Total}}}$$

5.0 Advanced Text Pre-Processing

To pre-process your text simply means to bring your text into a form that is predictable and analysable for your task.

There are different ways to pre-process your text. Here are some of the approaches that we should know about and we will try to highlight the importance of each.

The steps we will apply for advanced text pre-processing are:

1. Removing Punctuation
2. Removing Digits
3. Removing Stopwords
4. Changing to Lower-case words
5. Stemming or Lemmatization

5.0.1 Removing Punctuation

It helps to get rid of unhelpful parts of the data, or noise, by removing punctuations marks

5.0.2 Removing Digits

Since we are dealing with text, so the number might not add much information to text processing. So, numbers can be removed from the text.

5.0.3 Removing Stopwords

Stop words are a set of commonly used words in a language. Examples of stop words in English are "a", "the", "is", "are" etc. The intuition behind using stop words is that, by removing low information words from the text, we can focus on the important words instead.

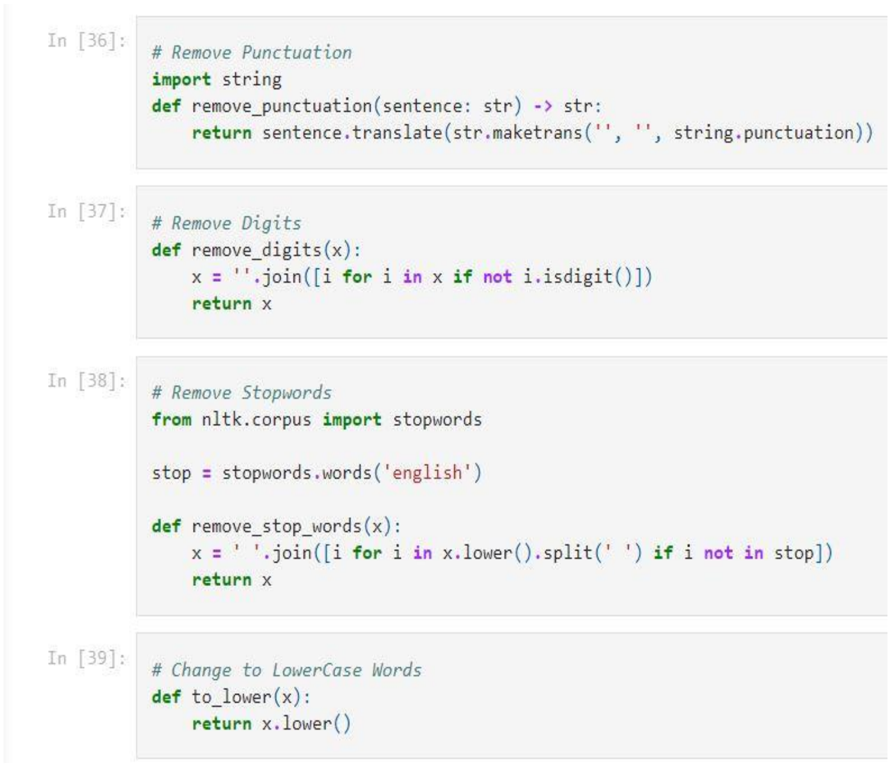
5.0.4 Changing to Lower-case words

Lowercasing your text data, although commonly overlooked, is one of the simplest and most effective forms of text pre-processing. It applies to most text mining and NLP problems and can help in cases where your dataset is not very large and significantly helps with the consistency of expected output.

5.0.5 Stemming or Lemmatization

Stemming is useful for dealing with sparsity issues as well as standardizing vocabulary. The idea is that, if say you search for "deep learning classes", you also want to surface documents that mention "deep learning class" as well as "deep learn classes", although the latter does not sound right. But you get where we are going with this. You want to match all variations of a word to bring up the most relevant documents.

Lemmatization on the surface is very similar to stemming, where the goal is to remove inflexions and map a word to its root form. The only difference is that lemmatization tries to do it the proper way. It does not just chop things off, it transforms words to the actual root. For example, the word "better" would map to "good". It may use a dictionary such as WordNet for mappings or some special rule-based approaches. Here is an example of lemmatization in action using a WordNet-based approach:

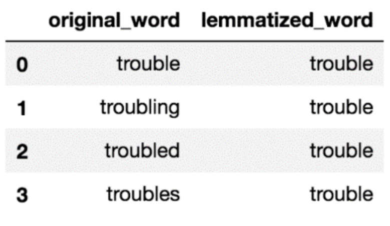


```
In [36]:
# Remove Punctuation
import string
def remove_punctuation(sentence: str) -> str:
    return sentence.translate(str.maketrans('', '', string.punctuation))
```

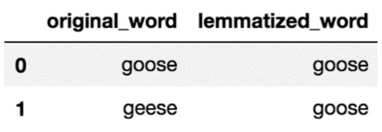
```
In [37]:
# Remove Digits
def remove_digits(x):
    x = ''.join([i for i in x if not i.isdigit()])
    return x
```

```
In [38]:
# Remove Stopwords
from nltk.corpus import stopwords

stop = stopwords.words('english')

def remove_stop_words(x):
    x = ' '.join([i for i in x.lower().split(' ') if i not in stop])
    return x
```

```
In [39]:
# Change to LowerCase Words
def to_lower(x):
    return x.lower()
```



	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble



	original_word	lemmatized_word
0	goose	goose
1	geese	goose

5.1 Count Vectorizer

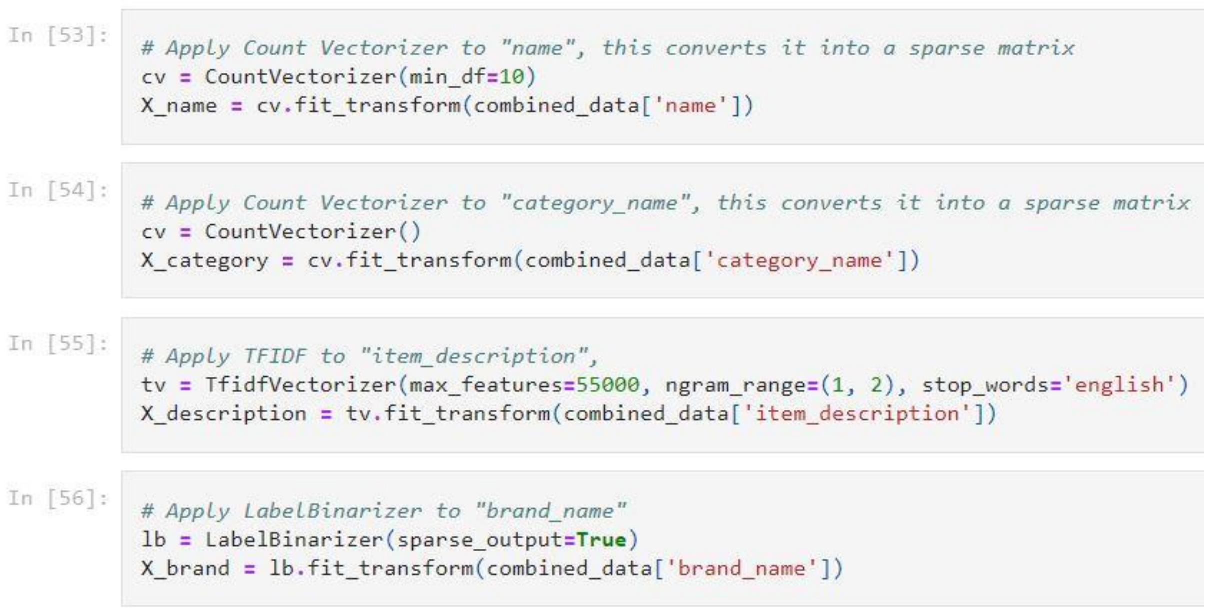
Convert a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts using scipy.sparse.csr_matrix. If you do not provide an a-priori dictionary and you do not use an analyser that does some kind of feature selection, then the number of features will be equal to the vocabulary size found by analysing the data.

5.2 TFIDF

Transform a count matrix to a normalized tf or tf-idf representation. Tf means term-frequency while tf-idf means term-frequency times inverse document-frequency. This is a common term weighting scheme in information retrieval, that has also found good use in document classification. The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

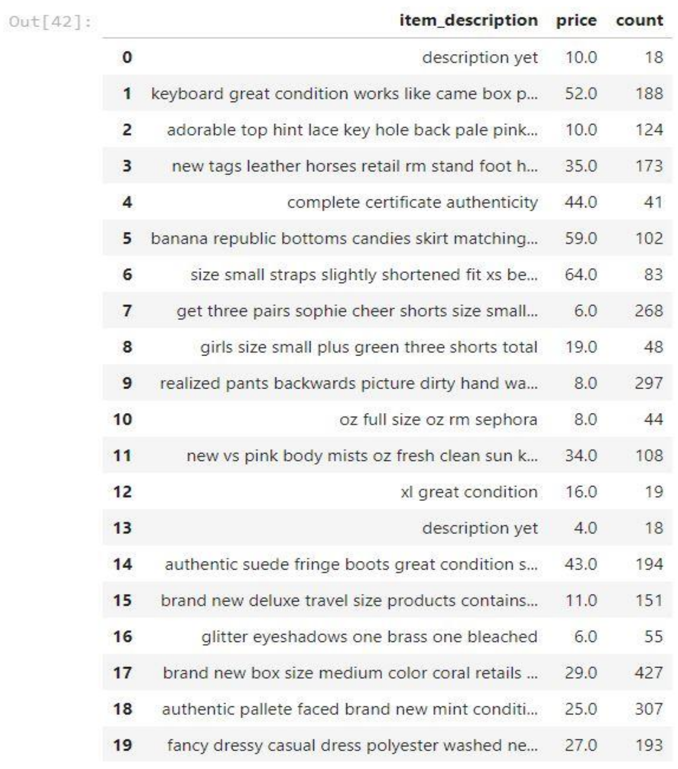
5.3 LabelBinarizer

Binarize labels in a one-vs-all fashion. Several regression and binary classification algorithms are available in scikit-learn. A simple way to extend these algorithms to the multi-class classification case is to use the so-called one-vs-all scheme. At learning time, this simply consists of learning one regressor or binary classifier per class. In doing so, one needs to convert multi-class labels to binary labels (belong or does not belong to the class). LabelBinarizer makes this process easy with the transform method. At prediction time, one assigns the class for which the corresponding model gave the greatest confidence. LabelBinarizer makes this easy with the inverse_transform method.



```
In [53]: # Apply Count Vectorizer to "name", this converts it into a sparse matrix
         cv = CountVectorizer(min_df=10)
         X_name = cv.fit_transform(combined_data['name'])

In [54]: # Apply Count Vectorizer to "category_name", this converts it into a sparse matrix
         cv = CountVectorizer()
         X_category = cv.fit_transform(combined_data['category_name'])

In [55]: # Apply TFIDF to "item_description",
         tv = TfidfVectorizer(max_features=55000, ngram_range=(1, 2), stop_words='english')
         X_description = tv.fit_transform(combined_data['item_description'])

In [56]: # Apply LabelBinarizer to "brand_name"
         lb = LabelBinarizer(sparse_output=True)
         X_brand = lb.fit_transform(combined_data['brand_name'])
```

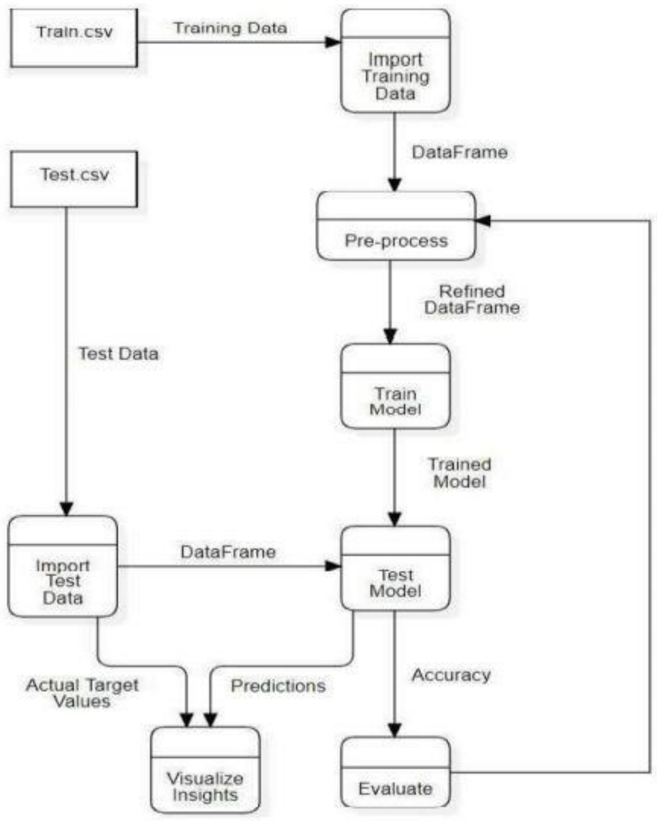
5.4 Data After Pre-Processing



Out[42]:

	item_description	price	count
0	description yet	10.0	18
1	keyboard great condition works like came box p...	52.0	188
2	adorable top hint lace key hole back pale pink...	10.0	124
3	new tags leather horses retail rm stand foot h...	35.0	173
4	complete certificate authenticity	44.0	41
5	banana republic bottoms candies skirt matching...	59.0	102
6	size small straps slightly shortened fit xs be...	64.0	83
7	get three pairs sophie cheer shorts size small...	6.0	268
8	girls size small plus green three shorts total	19.0	48
9	realized pants backwards picture dirty hand wa...	8.0	297
10	oz full size oz rm sephora	8.0	44
11	new vs pink body mists oz fresh clean sun k...	34.0	108
12	xl great condition	16.0	19
13	description yet	4.0	18
14	authentic suede fringe boots great condition s...	43.0	194
15	brand new deluxe travel size products contains...	11.0	151
16	glitter eyeshadows one brass one bleached	6.0	55
17	brand new box size medium color coral retails ...	29.0	427
18	authentic pallete faced brand new mint conditi...	25.0	307
19	fancy dressy casual dress polyester washed ne...	27.0	193

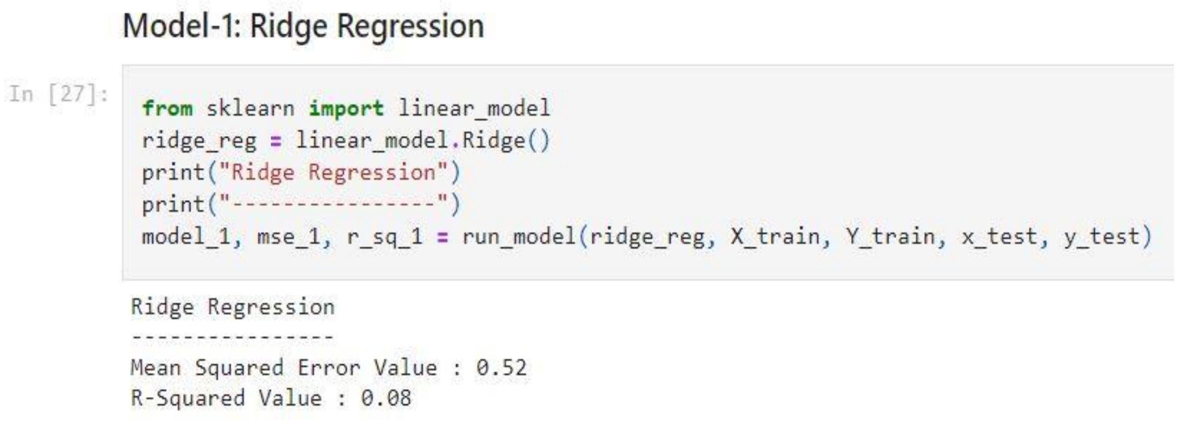
5.5 Dataflow diagram



6.0 Results and Observations

6.0.1 Without using advanced text pre-processing.

Ridge Algorithm



Model-1: Ridge Regression

```
In [27]:
from sklearn import linear_model
ridge_reg = linear_model.Ridge()
print("Ridge Regression")
print("----------------")
model_1, mse_1, r_sq_1 = run_model(ridge_reg, X_train, Y_train, x_test, y_test)
```

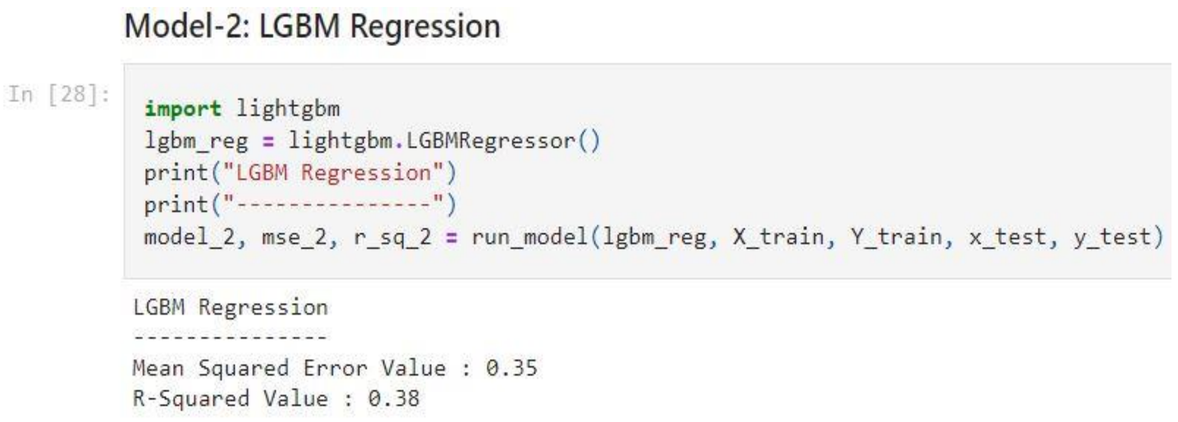
```
Ridge Regression
----------------
Mean Squared Error Value : 0.52
R-Squared Value : 0.08
```

Very high **MSE** (52%) and too low R^2 value (only 8%) for Ridge regression.

RMSE= Sq. Root(MSE) =0.72

Efficiency = 1 - 0.72 (RMSE) = 0.28 or 28% efficiency

LGBM Algorithm



Model-2: LGBM Regression

```
In [28]:
import lightgbm
lgbm_reg = lightgbm.LGBMRegressor()
print("LGBM Regression")
print("---------------")
model_2, mse_2, r_sq_2 = run_model(lgbm_reg, X_train, Y_train, x_test, y_test)
```

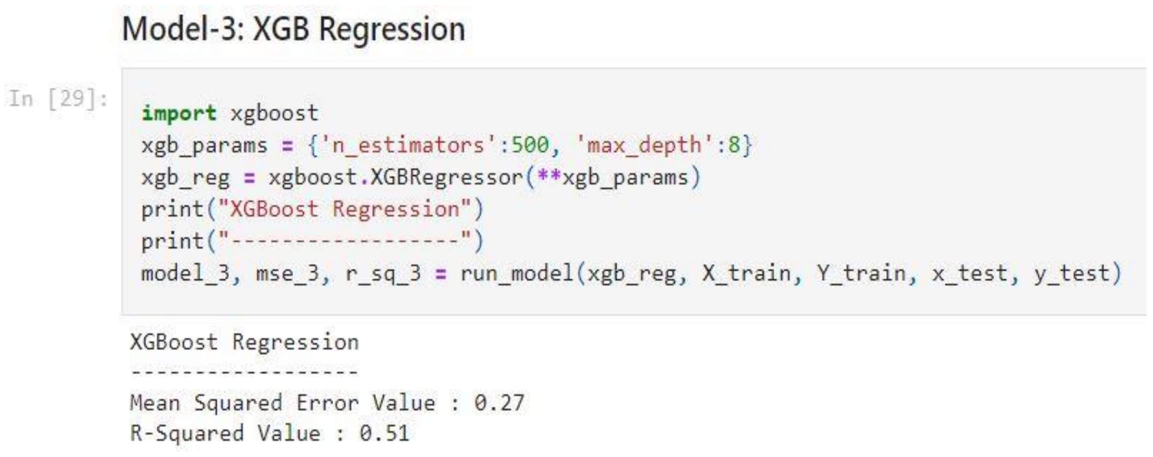
```
LGBM Regression
---------------
Mean Squared Error Value : 0.35
R-Squared Value : 0.38
```

High **MSE** (35%) and low R^2 value (39%) for LGBM regression.

RMSE= Sq. Root(MSE) =0.59

Efficiency = 1 - 0.59 (RMSE) = 0.41 or 41% efficiency.

XGB Algorithm



Model-3: XGB Regression

```
In [29]:
import xgboost
xgb_params = {'n_estimators':500, 'max_depth':8}
xgb_reg = xgboost.XGBRegressor(**xgb_params)
print("XGBoost Regression")
print("------------------")
model_3, mse_3, r_sq_3 = run_model(xgb_reg, X_train, Y_train, x_test, y_test)
```

```
XGBoost Regression
------------------
Mean Squared Error Value : 0.27
R-Squared Value : 0.51
```

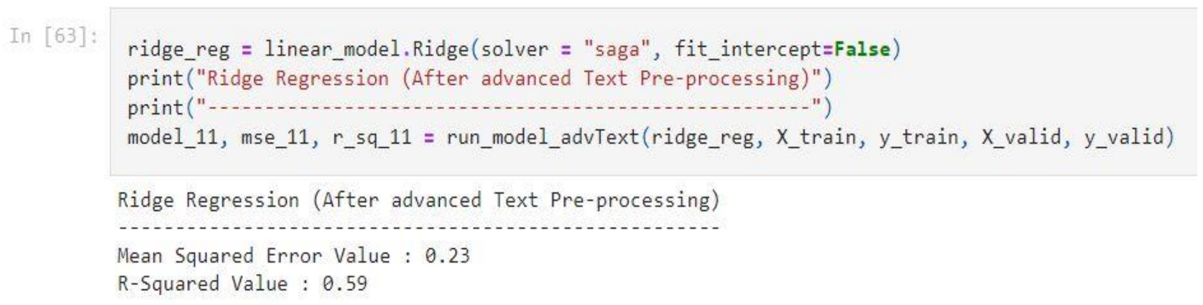
Medium **MSE** (27%) and Medium R^2 value (52%) for XGB regression.

RMSE= Sq. Root(MSE) =0.51

Efficiency = 1 - 0.51 (RMSE) = 0.49 or 49% efficiency.

6.0.2 With advanced text pre-processing.

Ridge Algorithm



```
In [63]:
ridge_reg = linear_model.Ridge(solver = "saga", fit_intercept=False)
print("Ridge Regression (After advanced Text Pre-processing)")
print("-----------------------------------------------------")
model_11, mse_11, r_sq_11 = run_model_advText(ridge_reg, X_train, y_train, X_valid, y_valid)
```

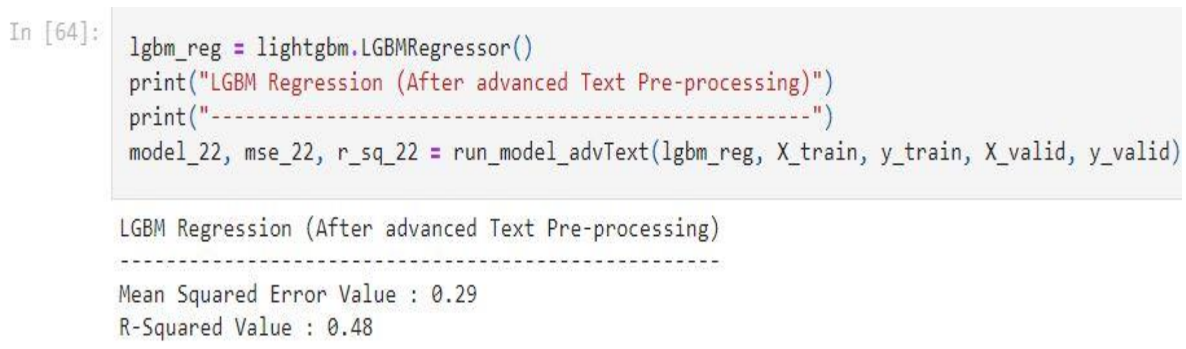
```
Ridge Regression (After advanced Text Pre-processing)
-----------------------------------------------------
Mean Squared Error Value : 0.23
R-Squared Value : 0.59
```

Very low **MSE** (23%) and medium R^2 value of 59% for Ridge regression.

RMSE= Sq. Root(MSE) =0.47

Efficiency = 1 - 0.47 (RMSE) = 0.52 or 52% efficiency

LGBM Algorithm



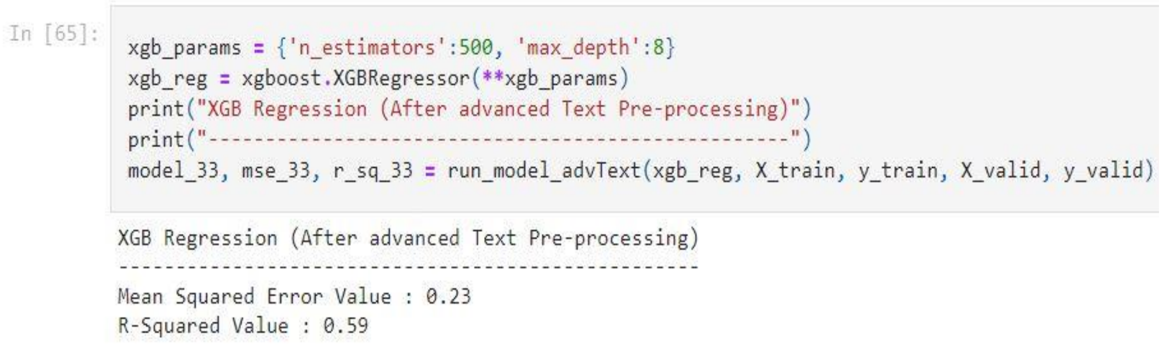
```
In [64]:  lgbm_reg = lightgbm.LGBMRegressor()
          print("LGBM Regression (After advanced Text Pre-processing)")
          print("---------------------------------------------------")
          model_22, mse_22, r_sq_22 = run_model_advText(lgbm_reg, X_train, y_train, X_valid, y_valid)

LGBM Regression (After advanced Text Pre-processing)
---------------------------------------------------
Mean Squared Error Value : 0.29
R-Squared Value : 0.48
```

Medium **MSE** (29%) and medium R^2 value (48%) for LGBM regression.

RMSE= Sq. Root(MSE) =0.53

Efficiency = 1 - 0.53 (RMSE) = 0.47 or 47% efficiency.

XGB Algorithm



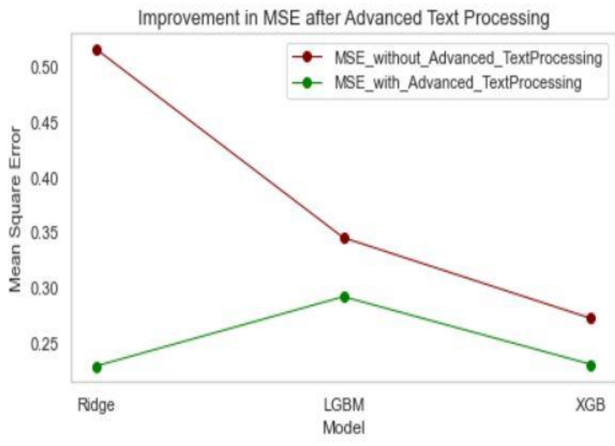
```
In [65]:  xgb_params = {'n_estimators':500, 'max_depth':8}
          xgb_reg = xgboost.XGBRegressor(**xgb_params)
          print("XGB Regression (After advanced Text Pre-processing)")
          print("---------------------------------------------------")
          model_33, mse_33, r_sq_33 = run_model_advText(xgb_reg, X_train, y_train, X_valid, y_valid)

XGB Regression (After advanced Text Pre-processing)
---------------------------------------------------
Mean Squared Error Value : 0.23
R-Squared Value : 0.59
```

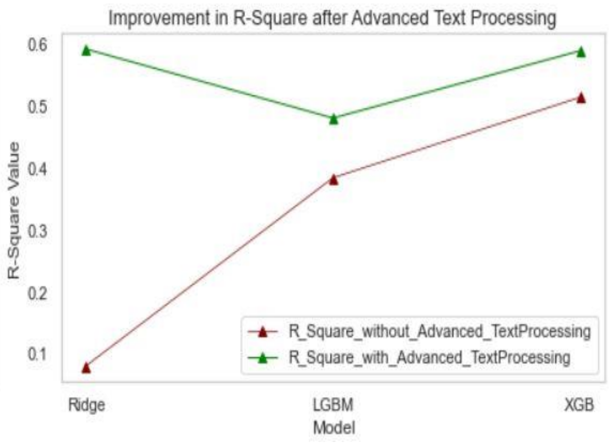
Very low MSE (23%) and medium R2 value of 59% for Ridge regression

RMSE= Sq. Root(MSE) =0.47

Efficiency = 1 - 0.47 (RMSE) = 0.52 or 52% efficiency

6.1 Result Comparison





- **MSE** phenomenally decreased for all three models after using advanced text pre-processing.
- **R^2** value also has now increased for all 3 models.

However, for the ridge model, the improvement seems to be the highest. Hence, we will choose the Ridge model to apply to the test data set to check for results.

The best outcome will come when we use the Advanced Text Pre-Processing method on the Ridge Regression model.

7.0 CONCLUSION

In this paper, we have implemented prediction for e-commerce websites selling household products. Considering the scale of the information the algorithms used are:

Three algorithms ridge regression, light-gradient boosting machine (LGBM), and XGBoost and NLP techniques using advanced text pre-processing, Count Vectorizer/ Tfidfvectorizer/ LabelBinarizer and applying it to the predictive models and compare the results and concluded the best model of the stated three models.

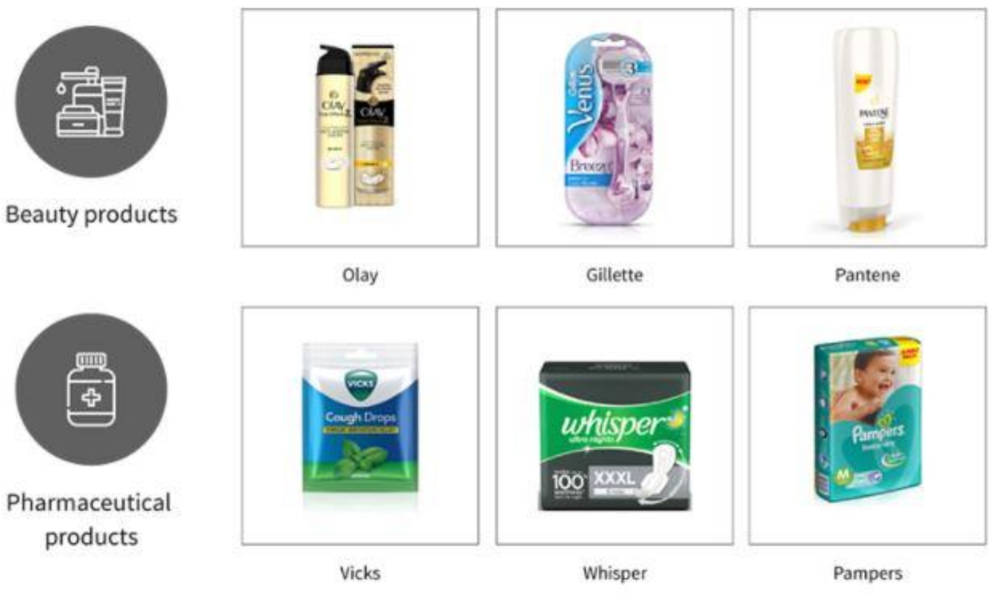
In Conclusion, we would like to state that out of the various models tested the Ridge Regression with Advanced Text Pre-Processing

7.1 FUTURE SCOPE

The pictures of the product may be enclosed as a feature for making the prediction model a lot stronger. Further, adding photographs of the condition of the items may accurately judge the price and authenticity of the product. Moreover, the verification of the seller's profile may be used as another feature for raising the performance of the model.

7.1.1 Image Classification

For increased accuracy, image classification using CNN is most effective. First and foremost, your solution will need a set of images. In this case, images of beauty and pharmacy products can be used as the initial training data set. The most common image data input parameters are the number of images, image dimensions, number of channels, and number of levels per pixel.



BIBLIOGRAPHY

[1]Sameer Chand Pudaruth. "Predicting the Price of Used Cars using Machine Learning Techniques", International Journal of Information & Computation Technology. ISSN 0974-2239 Volume 4, Number 7 (2014), pp. 753764

[2] Shonda Kuiper, "Introduction to Multiple Regression: How Much Is Your Car Worth? ", Journal of Statistics Education · November 2008

[3] Mariana Listiani, 2009. "Support Vector Regression Analysis for Price Prediction in a Car Leasing Application". Master Thesis. Hamburg University of Technology

REFERENCES

To conduct this project the following tools have been used.

1. **Python 3.9.5**
2. **Pandas (library)**
   - ❖ http://pandas.pydata.org/
3. **NumPy (library)**

- ❖ http://www.numpy.org/
- ❖ https://numpy.org/doc/stable/reference/generated/numpy.log1p.html

4. **Scikit-learn (library)**

- ❖ https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- ❖ https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html
- ❖ https://scikit-learn.org/stable/modules/linear_model.html

5. **Seaborn (library)**

- ❖ https://seaborn.pydata.org/index.html
- ❖ https://seaborn.pydata.org/generated/seaborn.heatmap.html

6. **SciPy (library)**

- ❖ https://docs.scipy.org/doc/scipy/reference/sparse.html

7. **nltk (library)**

- ❖ http://www.nltk.org/howto/

The following models were used to implement the data and to get the most accurate results:

1. **Ridge Regression or Linear model**

- ❖ https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

2. **Light Gradient Boosting Machine (LGBM)**

- ❖ https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMRegressor.html
- ❖ https://lightgbm.readthedocs.io/en/latest/Parameters.html

3. **Extreme Gradient Boosting Machine (XGBM)**

- ❖ https://xgboost.readthedocs.io/en/latest/parameter.html

The following data in the links proved to be very helpful for us in the completion of the project:

- ❖ https://towardsdatascience.com/ridge-regression-python-example-f015345d936b
- ❖ https://towardsdatascience.com/predicting-used-car-prices-with-machine-learning-techniques-8a9d8313952
- ❖ https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
- ❖ https://machinelearningmastery.com/clean-text-machine-learning-python/
- ❖ https://www.kdnuggets.com/2019/04/text-preprocessing-nlp-machine-learning.html
- ❖ https://en.wikipedia.org/wiki/Coefficient_of_determination
- ❖ https://www.geeksforgeeks.org/python-mean-squared-error/
- ❖ https://blog.exploratory.io/a-practical-guide-of-exploratory-data-analysis-with-linear-regression-part-1-9f3a182d7a92