A Data Science Case Study with Python: Mercari Price Prediction

Lecturer: Joko Triloka, Ph.D

The most effective way to learn data science is by solving data science related problems. Reading, listening and taking notes is valuable, but once you work through a problem, concepts solidify from abstractions into tools you feel confident using. Generally a full cycle data science project includes the following stages:

1. Data Gathering & Wrangling
2. Data Analysis & Modeling
3. Communication & Deployment

In this case study, we will walk through the **Analysis, Modelling and Communication** part of the workflow. The general steps involved for solving a data science problem are as follows:

1. Understand the problem and data
2. Data exploration / data cleaning
3. Feature engineering / feature selection
4. Establishing a performance metric
5. Model evaluation and selection
6. Model optimization
7. Draw conclusions and document work

Those of you who are not familiar with the field of Data Science and Python programming language can still follow through the article as it will give a high level overview about how these kind of problems are approached and solved.

While some code snippets are included within this, for the full code you can check out this Jupyter Notebook.

Problem Definition

Mercari is Japan's biggest community powered shopping app where people can sell and buy a variety of brand new and used products of different brands, from sweaters to smartphones. Now Mercari would like to suggest the correct prices to the sellers but this is tough because their sellers are enabled to put just about anything, or any bundle of things on Mercari's marketplace.

So our objective is to build a model that automatically suggests the right product prices to the sellers. We are provided with the following information for each product:

train_id — the id of the product

name — the title of the product

item_condition_id — the condition of the product provided by the sellers

category_name — category of the product

brand_name — the product's brand name

shipping — 1 if shipping fee is paid by seller and 0 if shipping fee is paid by buyer

item_description — the full description of the product

price — the price that the product was sold for (**This is the target variable that we will predict**)

This type of problem lies under the category of Supervised Regression Machine Learning:

- **Supervised:** We have access to both the features and the target and our goal is to train a model that can learn a mapping between the two.
- **Regression:** The target variable, price, is a continuous variable.

Exploratory Data Analysis (EDA)

EDA is an open-ended process where we calculate statistics and make figures to find trends, anomalies, patterns, or relationships within the data. In short, the goal of EDA is to learn what our data can tell us. It generally starts out with a high level overview, then narrows in to specific areas as we find interesting parts of the data. The findings may be interesting in their own right, or they can be used to inform our modeling choices, such as by helping us to decide which features to use. In a hurry to get to the machine learning stage, some data scientists either entirely skip the exploratory process or do a very perfunctory job but in reality EDA is one of the most crucial steps in solving a data science related problem.

The data set can be downloaded from Kaggle. To validate the result, we only need the 'train.tsv' data file. So let's get started!

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.model_selection import train_test_split
```

We will split the data into train and test sets in the proportion of 80% and 20% respectively. As a common practice, we will conduct EDA on the train data set only

```
In:
data = pd.read_csv('train.tsv', sep='\t')train, test =
train_test_split(data, test_size=0.2)
print(train.shape, test.shape)Out:
(1186028, 8) (296507, 8)
```

There are around 1.18 million data points/rows in the train data and 0.296 million data points/rows in the test data. 8 is the number of columns/features in both the sets. Features and variables mean the same thing here, so they might be used interchangeably within this study.

```
#this command displays first few rows of the data set
train.head()
```

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

Check for missing values

A common problem when dealing with real-world data is missing values. These can arise for many reasons and have to be either filled in or removed before we train a machine learning model. First, let’s get a sense of how many missing values are in each column.

Your selected dataframe has 8 columns.
There are 3 columns that have missing values.

	Missing Values	% of Total Values
brand_name	632682	42.7
category_name	6327	0.4
item_description	4	0.0

The ‘brand_name’ feature has 42% missing values. Some of the approaches that are usually considered while dealing with missing values are:

1. Remove the records with the missing values.
2. Remove the feature itself if the number of missing values is higher than some threshold, say 50%.
3. Consider ‘missing values’ as another category of that respective feature itself.

In this project, we will go forward with the third approach.

Now we will start analyzing the features one by one. We will first go through the target variable, Price, and then start analyzing the predictor variables individually and also see how it interacts with the Price variable**.**

Price

Let’s check the distribution of the Price variable and go through some basic statistics.

```
train['price'].plot.hist(bins=50, figsize=(10,5),
edgecolor='white',range=[0,500])
plt.xlabel('Price', fontsize=15)
plt.ylabel('Frequency', fontsize=15)
plt.tick_params(labelsize=15)
plt.title('Price Distribution - Training Set', fontsize=17)
plt.show()
```


```
count    1.186028e+06
mean     2.672386e+01
std      3.844205e+01
min      0.000000e+00
25%      1.000000e+01
50%      1.700000e+01
75%      2.900000e+01
max      2.009000e+03
Name: price, dtype: float64
```

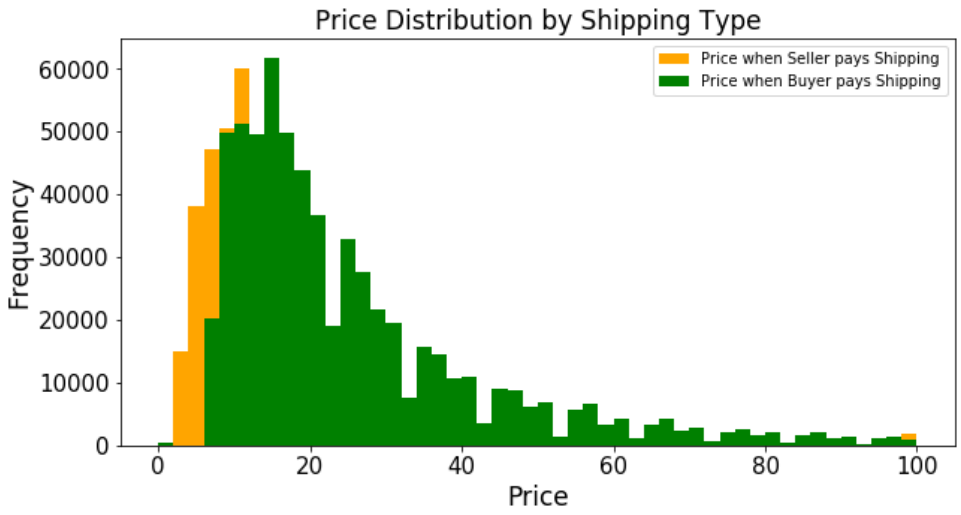
- The distribution of the Price variable aligns with the above statistics, i.e. the distribution is **right skewed.**
- The median price of the items is $17.
- 75% of the items have price below $29.
- The maximum price of an item from the data is $2009.

Shipping Fee

In:

```
train['shipping'].value_counts(normalize=True)*100Out:
0    55.267667
1    44.732333
Name: shipping, dtype: float64
```

The shipping fee for 55.26% of the items are paid by the buyers. Let's see how the shipping feature relates with price.



The median price of the products is $14.0 if seller pays for the shipping while the median price of the products is $20.0 if buyer pays for the shipping.

Normally when we buy products online, we need to pay for shipping or delivery of products which are below a certain price. But here the trend is kind of opposite since the median price of items for which the seller pays the shipping fees is lower than the median price of the items for which the buyer pays the shipping fees.

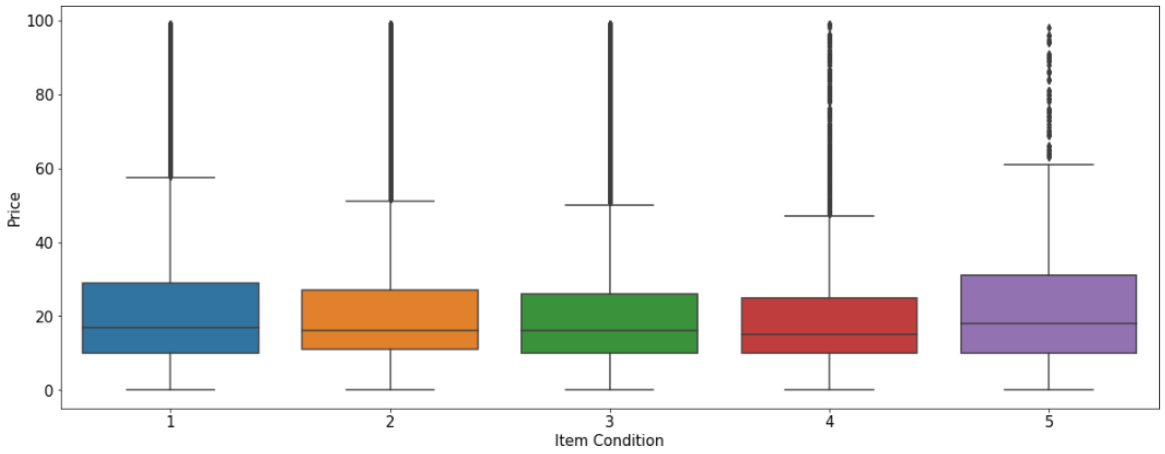
Item Condition

The 'item_condition_id' has five unique values ranging from 1 to 5. **Lower the number, better the condition of the item.**

```
In:
train['item_condition_id'].value_counts(normalize=True,
sort=False)*100Out:
1    43.194511
2    25.323264
3    29.166934
4     2.153153
5     0.162138
Name: item_condition_id, dtype: float64
```

43% of the items have item condition ID as 1 while only 0.16% of the items have item condition ID has 5. Let's see how this feature relates with Price. For comparing and visualizing the relation between a categorical variable and a numerical variable, Box-Plots are very helpful. The code below is used for plotting Box-Plots.

```
#for easier visualization, we are considering the prices from
range of 0-100
price_100 = train[train['price']<100]fig, ax =
plt.subplots(figsize=(20,7.5))
sns.boxplot(x='item_condition_id', y='price', data=price_100,
ax=ax)
plt.xlabel('Item Condition', fontsize=15)
plt.ylabel('Price', fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```



Though the proportion is low, items with **item condition ID as 5** have **higher median price** when compared with items having better condition.

After going through some of the data in which the item condition ID is given as 5, most of the products with this condition, especially electronic products are being sold for their parts which itself can prove to be valuable. This pretty much explains the higher median price.

Category of the Product

There are 1268 unique categories in the ‘category_name’

```
(train['category_name'].value_counts(normalize=True)*100).head(6)Out:
Women/Athletic Apparel/Pants, Tights, Leggings              4.060365
Women/Tops & Blouses/T-Shirts                               3.134841
Beauty/Makeup/Face                                          2.321684
Beauty/Makeup/Lips                                          2.025398
Electronics/Video Games & Consoles/Games                    1.798039
Beauty/Makeup/Eyes                                          1.720390
Name: category_name, dtype: float64
```

As we can see, for each item, there are three sets of categories separated by ‘/’. The categories are arranged from top to bottom with respect to the comprehensiveness. Therefore we can split the categories into three different columns. The three categories will signify main category, first subcategory and second subcategory. While splitting, the missing values will be filled with the string ‘Category Unknown’.

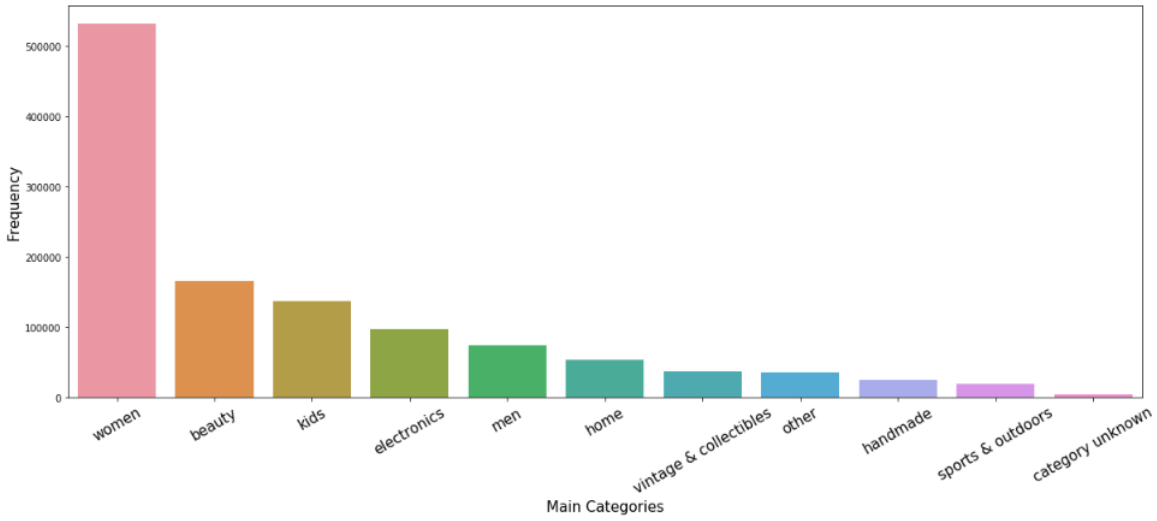
As a common preprocessing practice, we will convert all the textual categorical values to lower case. The reason for doing this will be clear with the following example:

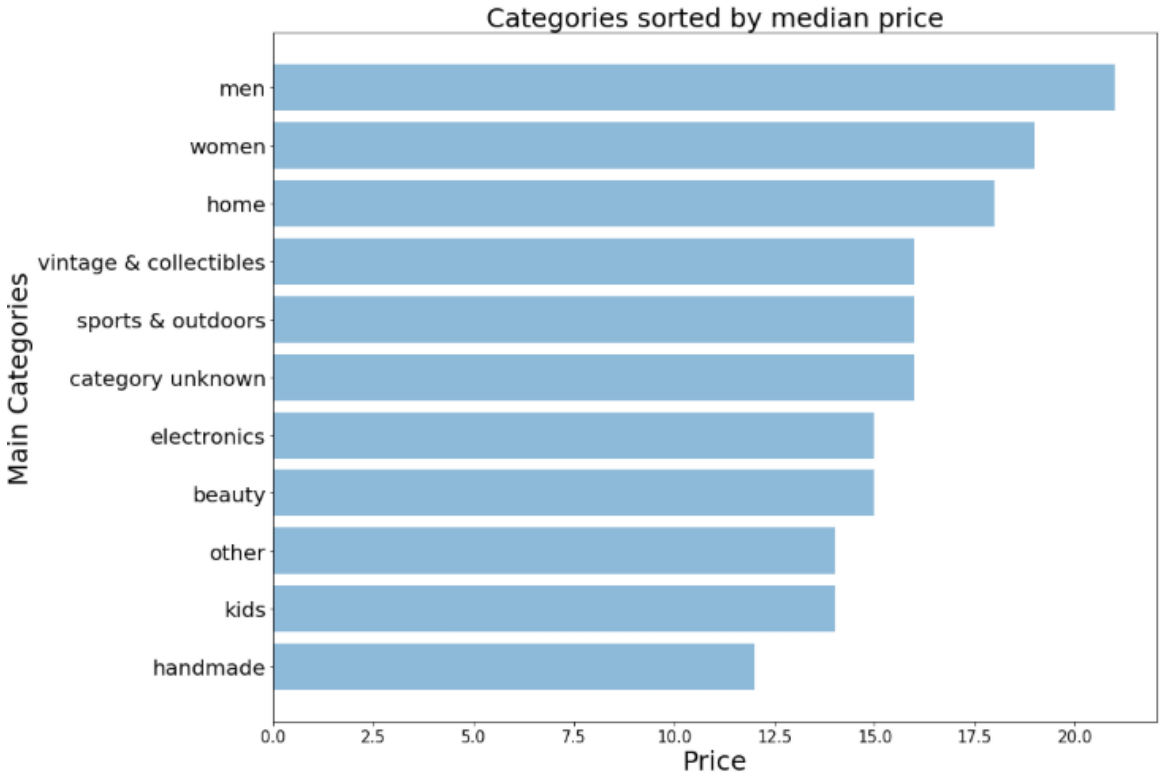
Suppose there are two rows in the data and the product belonging to both the rows is an iPhone and the brand name corresponding to the first entry is ‘APPLE’ and for the second entry it is given as ‘Apple’. So when we featurise the data for applying the machine learning models, any featurisation technique that we will apply will consider ‘APPLE’ and ‘Apple’ has two different values even though they mean the same thing. So, for our model to consider these two values as the same, we will convert all the textual categorical values to lower case. Effectively, ‘APPLE’ and ‘Apple’ will become ‘apple’ and ‘apple’.

After splitting the categories, we have **11 unique main categories, 114 unique first sub-categories, 863 unique second sub-categories.**

Main Category

Let’s see which one of the main categories have the highest number of products in the data and also see how the main categories stack up according to the median price of their respective products.



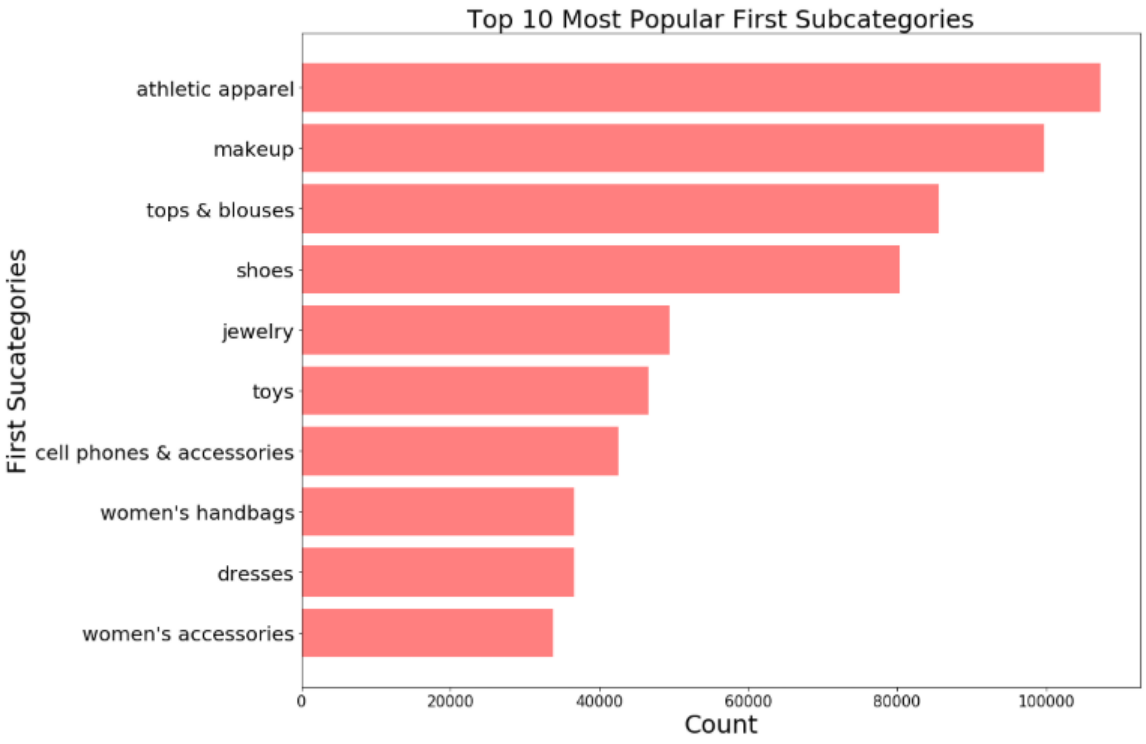


The maximum number of products, i.e. 44.8% of the total products belong to 'Women' category followed by 'Beauty' category products which takes up around 14% of the total products while 1.7% of the products, being the minimum, belong to the 'Sports and Outdoor' category.

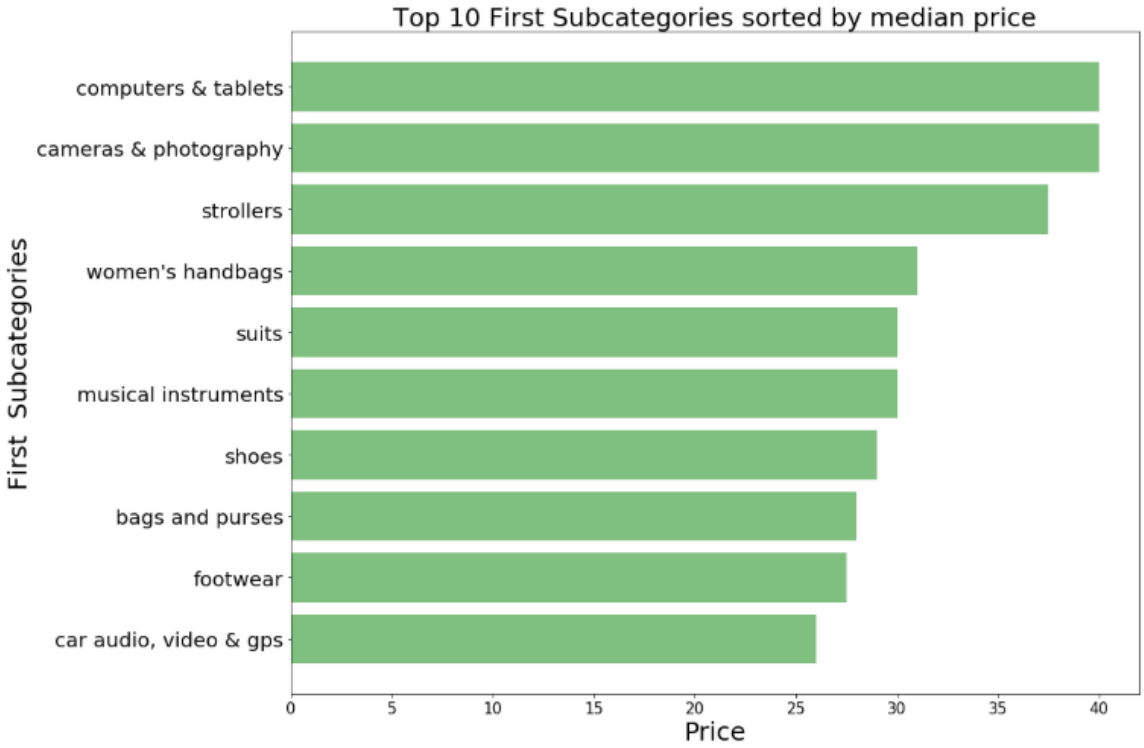
The highest median price of $21 belongs to the items from ‘Men’ category followed by ‘Women’ category having a median price of $19 while the items from ‘Handmade’ category have the lowest median price of $12.

First Subcategory

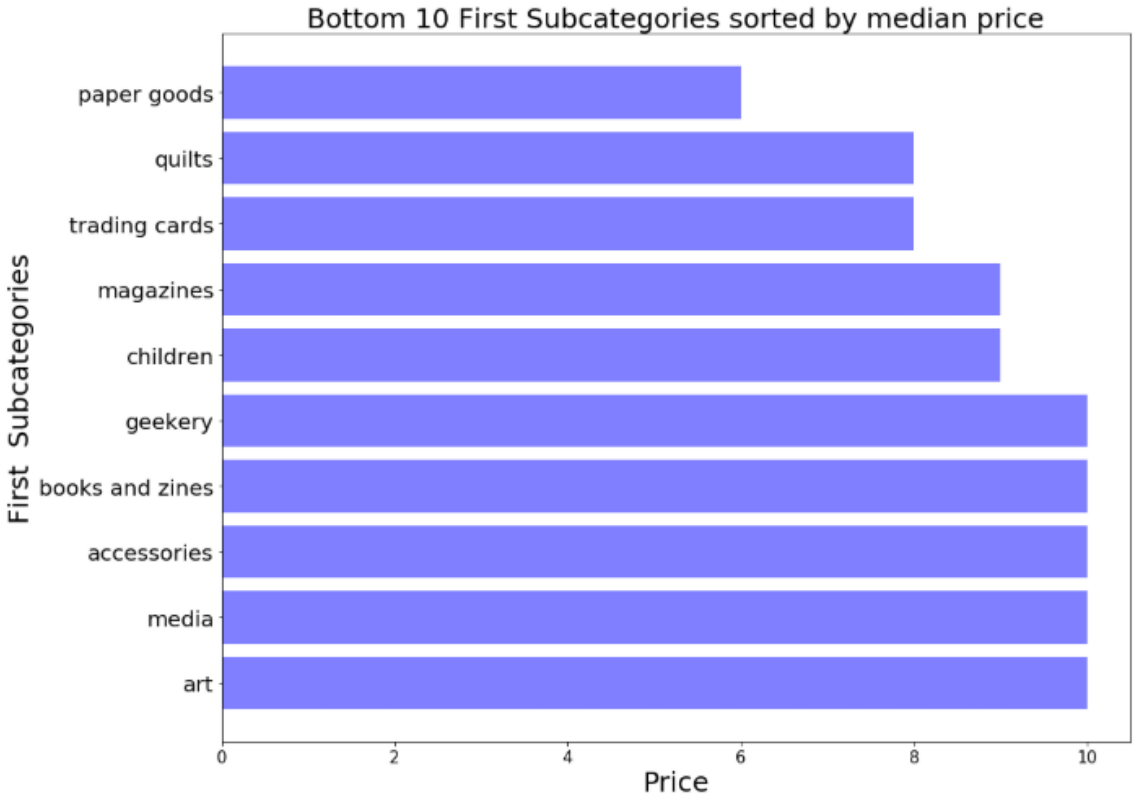
Since there are 114 unique subcategories, it will be difficult to visualize all of them. So, we will take a look at top 10 most popular subcategories and top & bottom 10 subcategories sorted according to the median prices of their respective items.



The most popular subcategory is ‘Athletic Apparel’ which aligns with the previous observation that the most popular main category is ‘Women’ and ‘Athletic Apparel’ comes under both the ‘Women’ and ‘Men’ categories. 9% of the total products comes under the Athletic Apparel category.



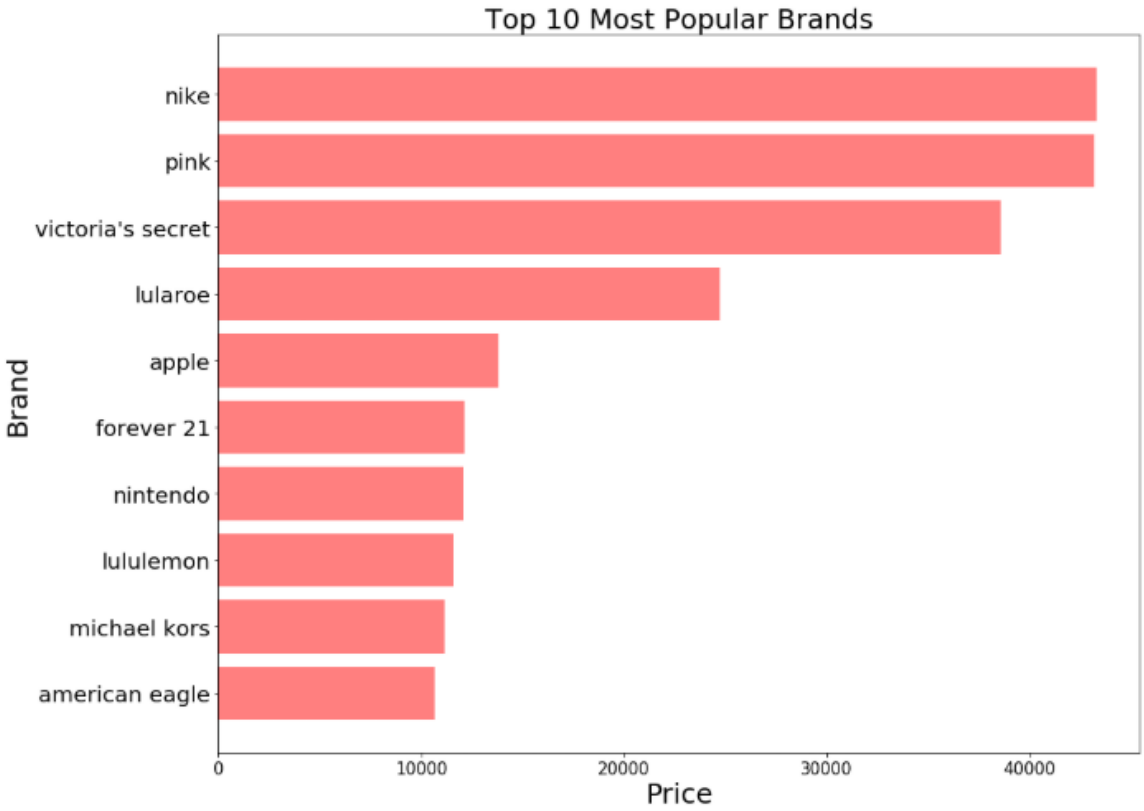
From the perspective of the median price of the items, the items from the subcategory of 'Computers & Tablets' have the highest median price, with the median price being $40.



The items from the subcategory ‘Paper Goods’ have the lowest median price of $6. ‘Paper Goods’ come under the ‘Handmade’ category. This also validates the previous observation that the items from ‘Handmade’ category have the lowest median price out of all the main categories.

Brand Name

There are **4535** unique brands in the data. Let’s take a look at the top 10 most popular brands.

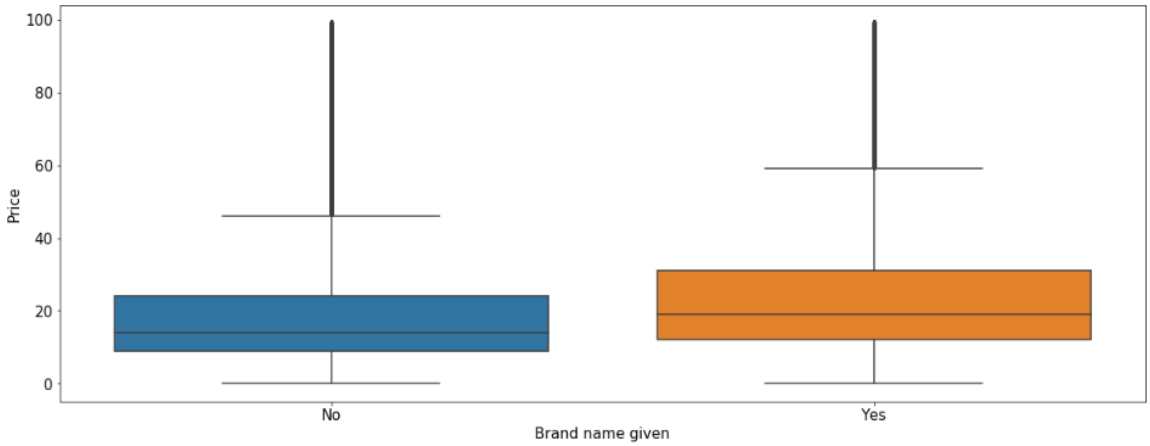


Nike and Pink are the top two most popular brands and 12.6% of the total products in the data belong to these two brands.

When we had checked for missing values, ‘brand_name’ feature had 42.7% missing values. So to deal with it, we will impute the missing values with the string ‘brand unavailable’.

```
train['brand_name'] = train['brand_name'].fillna('brand_unavailable')
```

Since there were 42% missing values, it will be interesting to see whether the price is impacted when the brand name is given or not for that particular product. For that, we will create a new feature named 'brand_name_given' with the values 'Yes' and 'No' denoting whether the brand name is given or not. Let's find out!



After looking at the box plots, although there is good amount of overlap, we can say that there is a considerable difference in the prices when the brand is given and when it's not given. The median price of the product when the brand name is given is $20 and when the brand name is not given, the median price is $14. In a nutshell, this feature will be helpful for the ML model to map some kind of pattern from it.

Item Description

The 'item description' feature falls under the category of unstructured text data. For this type of data, the preprocessing steps will include the following operations:

1. Convert all the words to lowercase.
2. Remove common words like 'or', 'and', 'is', 'the', 'was' etc. We remove them because these kinds of words will be present within the data in high proportion and the ML models won't be able to generate any meaningful patterns from it. These set of words are known as 'Stopwords'.
3. Remove special characters like '&', '@', '!', '?' etc.

For visualization, one of the most effective ways to visualize text data is by plotting ‘Word Clouds’. We will see in the following word clouds about how do the words in the item description compare when the price increases. To check that, we will sort the data according to the prices from low to high and then divide the data into four equivalent parts. By doing this the first quarter will have products with the lowest prices and consequently the fourth quarter will have products with the highest prices.

First Quarter

Second Quarter

Third Quarter

Fourth Quarter

All the four word clouds are almost identical to each other. Words like ‘brand new’, ‘never used’, ‘good condition’, ‘size’, ‘medium’ etc. are used in a high frequency for most of the products irrespective of the price. This shows that the sellers most of the time try to put in a good word for their product in the product description section, so that they don’t have much trouble in selling off the product quickly.

One interesting thing in the first word cloud are the words ‘description yet’. After a quick glance through the data, there are entries in which the description is not given but just the words ‘No description yet’ and there are 5.57% rows with such words. Just like we did with the ‘brand_name’ feature, we will verify whether the price of the product is impacted when the description is given or not.

There is not much difference to be seen here unlike the ‘brand_name’ feature. The median price of the product when the description is given and when it is not given are almost the same. The 75th percentile value of the product price when the description is given is 5$ more than the 75th percentile value of the product price when the description is not given.