Problem

Product pricing gets harder at scale, considering just how many products are sold online. Clothing has strong seasonal pricing trends and is heavily influenced by brand names, while electronics have fluctuating prices based on product specs.

Mercari, Japan's biggest community-powered shopping app, knows this problem deeply and they would like to offer pricing suggestions to sellers. This problem is challenging because sellers can put just about anything, or any bundle of things, on Mercari's marketplace.

The goal of the project is to build a model that automatically suggests the right product prices.

Data

The data is obtained from https://www.kaggle.com/c/mercari-price-suggestion-challenge/data and consists of the columns:

- train_id or test_id - the id of the listing
- name - the title of the listing. Note that we have cleaned the data to remove text that look like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]
- item_condition_id - the condition of the items provided by the seller
- category_name - category of the listing
- brand_name
- price - the price that the item was sold for. This is the target variable that you will predict. The unit is USD. This column doesn't exist in test.tsv since that is what you will predict.
- shipping - 1 if shipping fee is paid by seller and 0 by buyer
- item_description - the full description of the item. Note that we have cleaned the data to remove text that look like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]

Using the given data, the goal is to predict the sale price of a listing based on information (features) that a user provides for this listing.

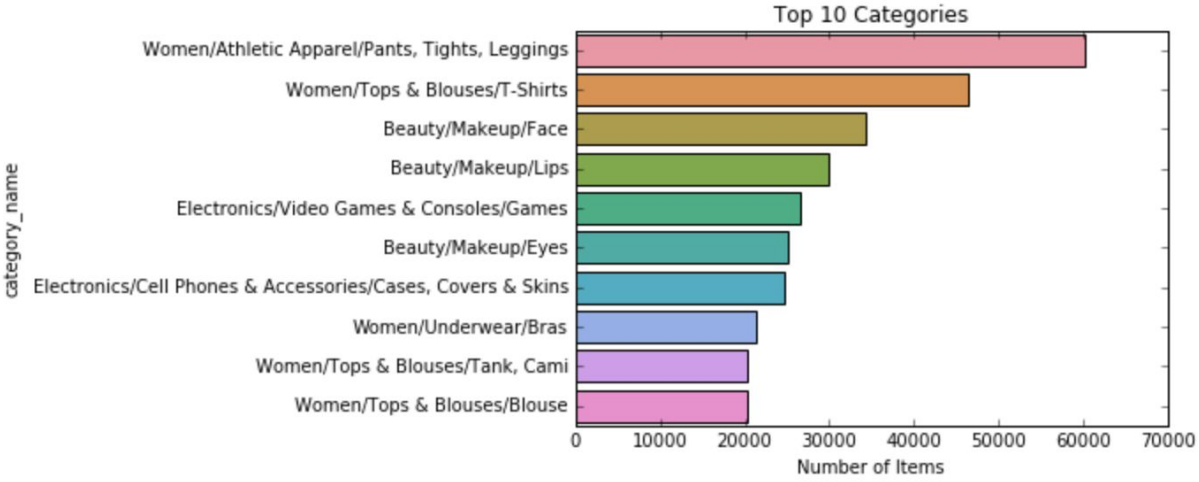
Since the dataset was obtained from Kaggle, the data is already pretty clean. There are some “NaN” values in category_name, brand_name, and item_description column which can be seen from the table below.

```
train_id                0
name                    0
item_condition_id        0
category_name         6327
brand_name          632682
price                    0
shipping                 0
item_description         4
dtype: int64
```

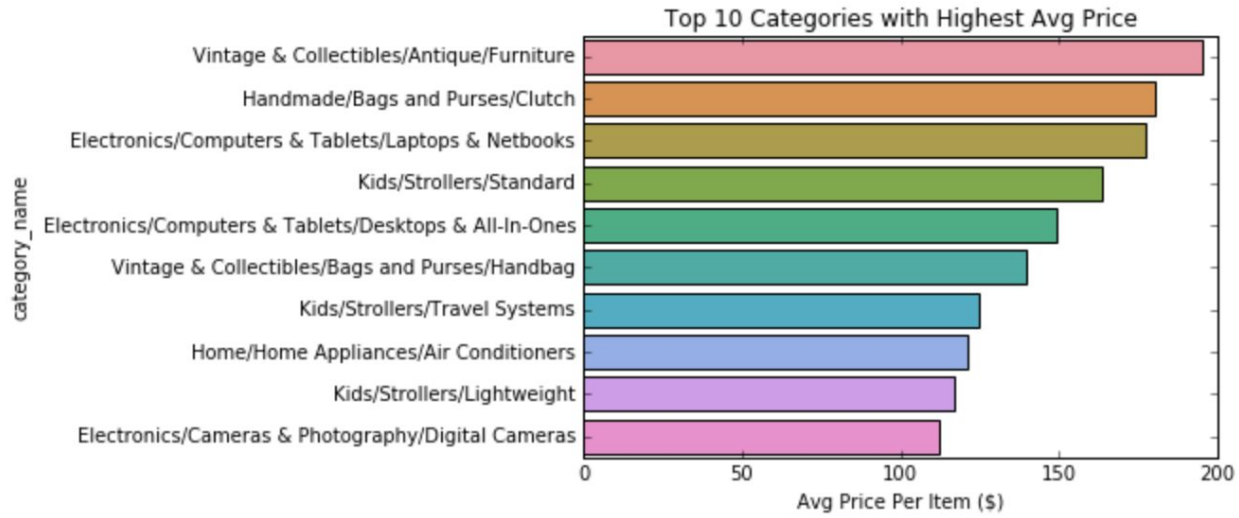
For the category_name column, the NaNs are replaced with ‘Other’. For the brand_name column, the NaNs are replaced with ‘Not Specified’. For the item_description column, the NaNs are replaced with ‘No description yet’.

Data Exploration

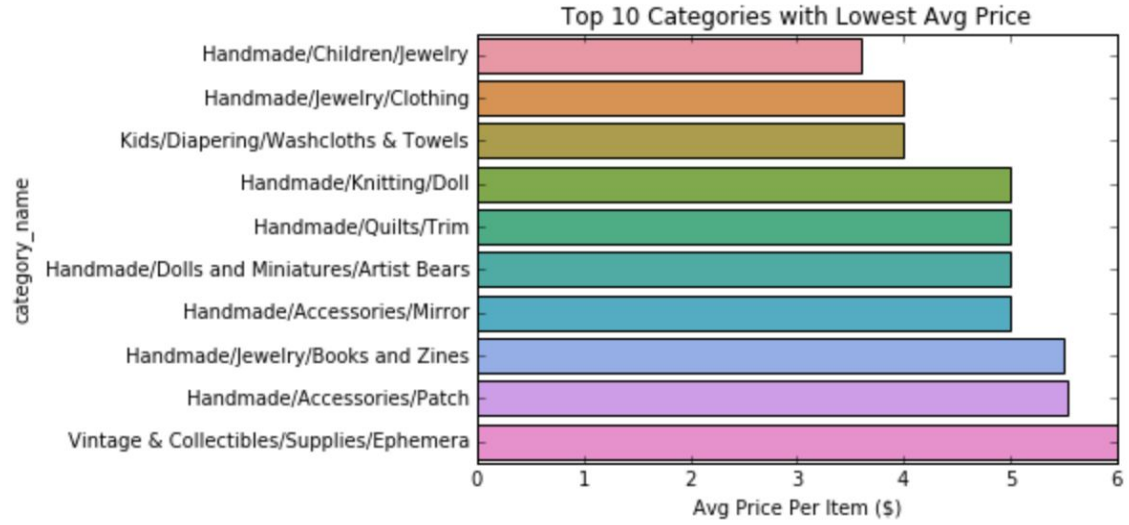
Analysis of Item Categories:
The item categories are 3 layers deep. For example: Men/Tops/T-shirt. The most popular category, as can be seen from the plot below, is Women/Athletic Apparel/Pants, Tights, Leggings. It looks like there are a lot more items for women than men. Among the top 10 categories, 8 of them are in 'Women' category and the other 2 are 'Electronics'.



Top 10 most expensive categories, as can be seen from the plot below, are as expected with furniture, electronics, handbag, and strollers dominating. The most expensive category is Vintage & Collectibles/Antique/Furniture which items in that category have an average price of almost $200.

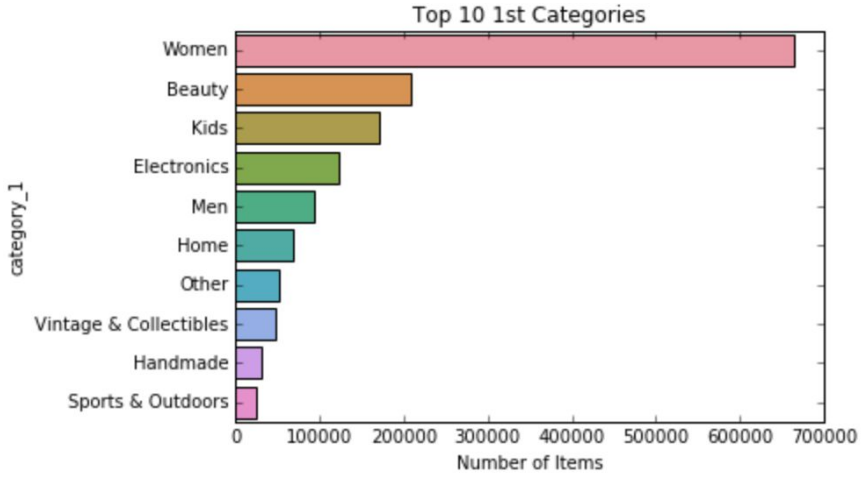


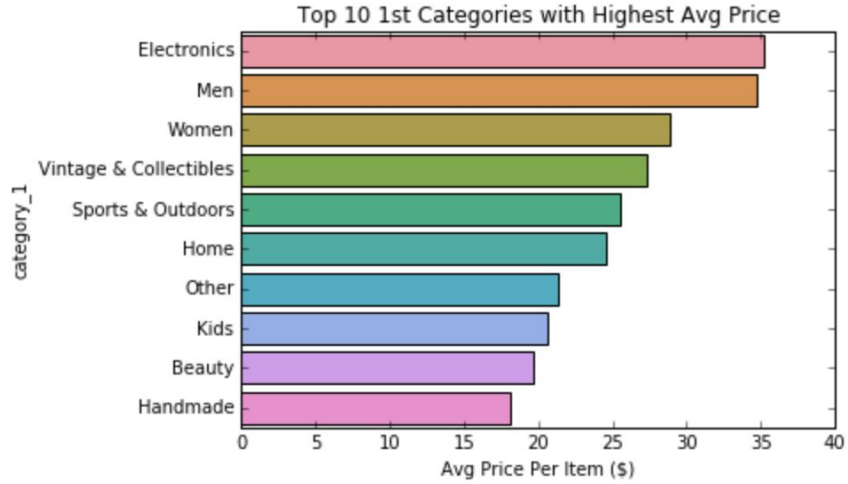
The Top 10 least expensive categories are mostly handmade items (including jewelery which is somewhat counter intuitive).

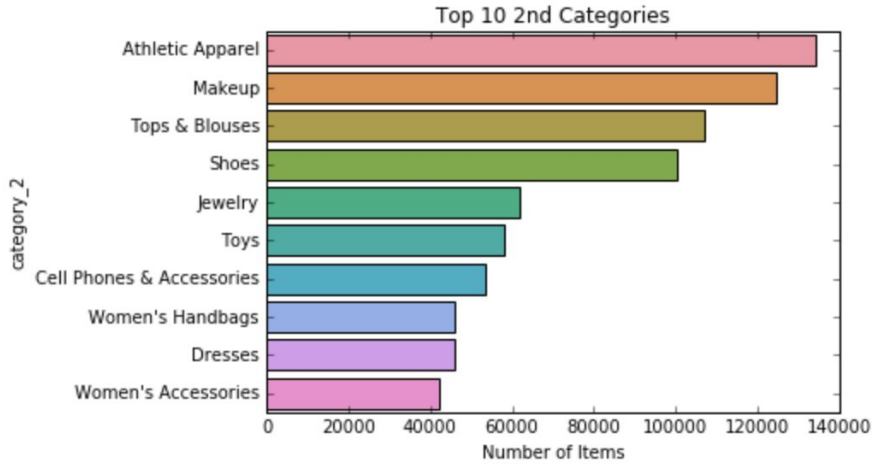


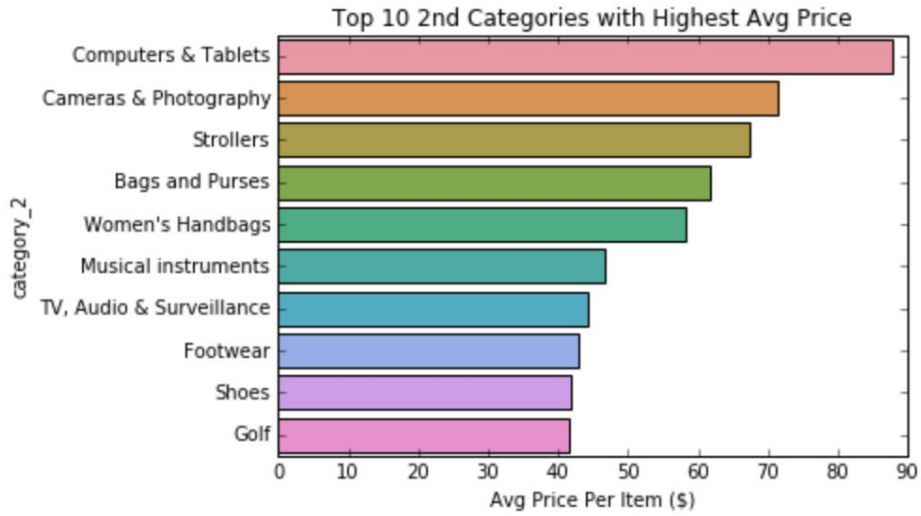
The categories then are split to main category and different subcategories. We will have category 1, category 1 2, category 2, and category 3. For the example, for Men/Tops/T-shirt: Men will be the 1st category, Tops the 2nd category, Men/Tops the 1_2 category, and T-shirt the 3rd category. There are 10 unique values for category 1, 113 unique values for category 2, and 870 unique values for category 3.

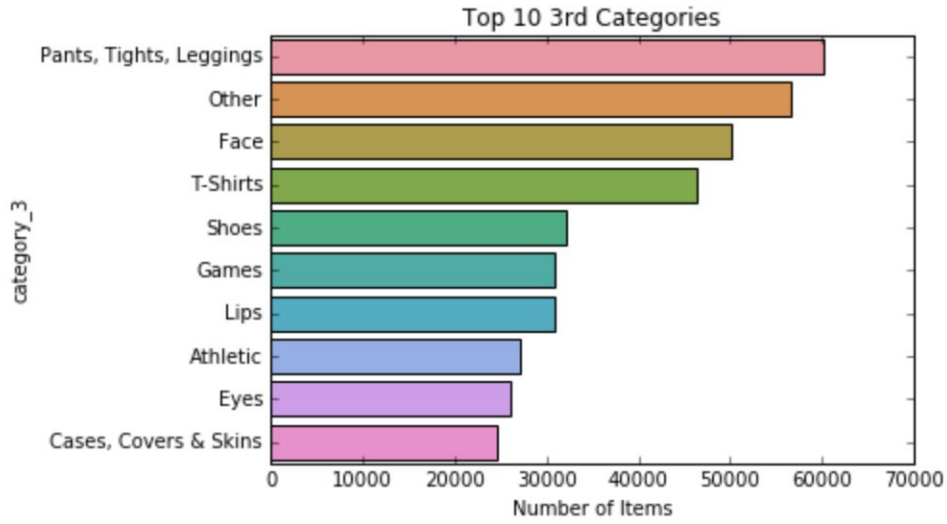
The plots below further explore these main and sub categories and are consistent with our previous observation. Women category has the most number of items. The most expensive categories are: Electronics, Men, and Women (because of luxury items as can be seen from the brands analysis in the next section).

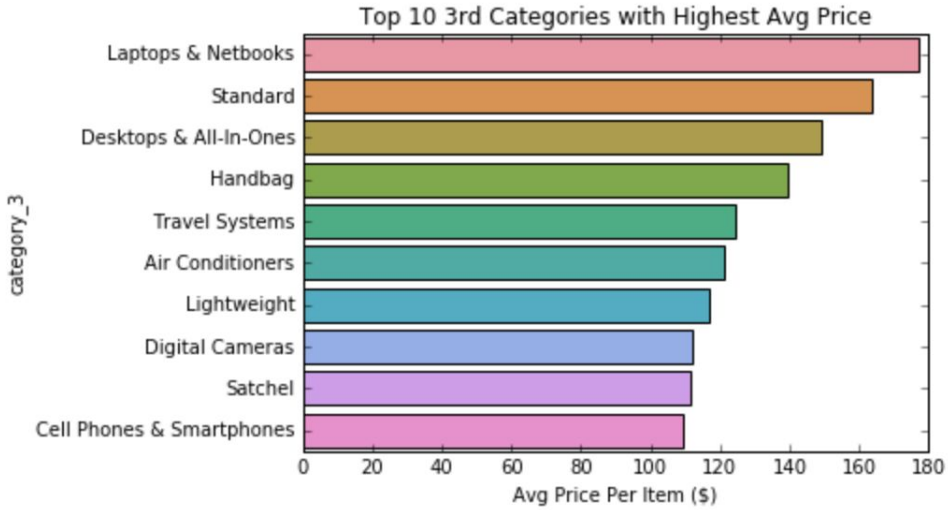




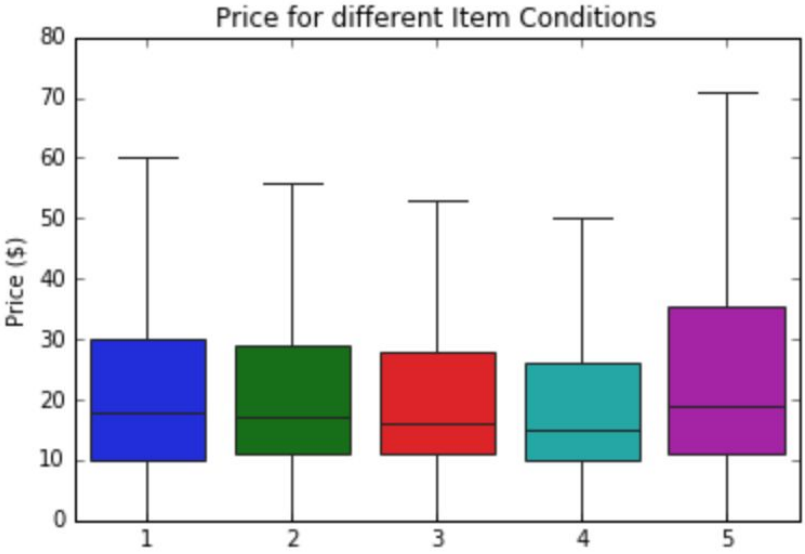


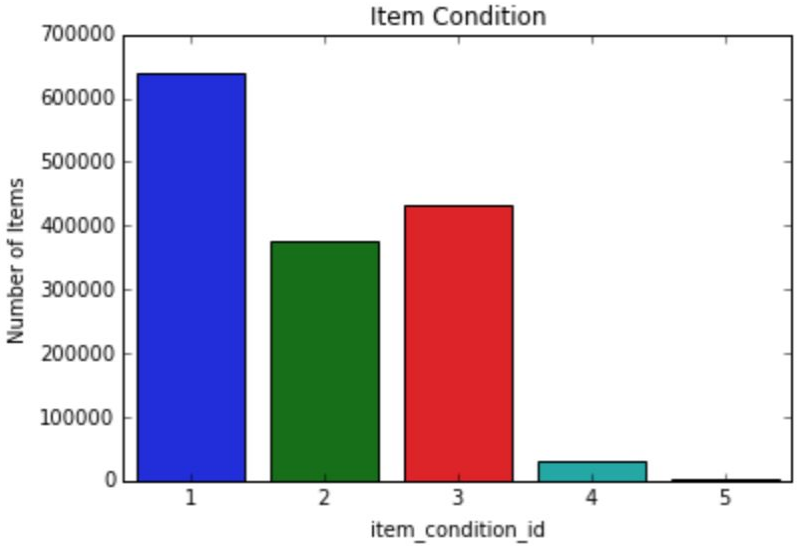

Top 10 2nd Categories with Highest Avg Price
Computers & Tablets
Cameras & Photography
Strollers
Bags and Purses
Women's Handbags
Musical instruments
TV, Audio & Surveillance
Footwear
Shoes
Golf
category_2
0
10
20
30
40
50
60
70
80
90
Avg Price Per Item ($)


Top 10 3rd Categories
Pants, Tights, Leggings
Other
Face
T-Shirts
Shoes
Games
Lips
Athletic
Eyes
Cases, Covers & Skins
category_3
0
10000
20000
30000
40000
50000
60000
70000
Number of Items


Top 10 3rd Categories with Highest Avg Price
Laptops & Netbooks
Standard
Desktops & All-In-Ones
Handbag
Travel Systems
Air Conditioners
Lightweight
Digital Cameras
Satchel
Cell Phones & Smartphones
category_3
0
20
40
60
80
100
120
140
160
180
Avg Price Per Item ($)

Analysis of Item Conditions:
The item conditions are not really explained by Mercari so we don't really know whether 1 represents 'new' quality or 'poor' quality. But, intuitively, the item condition should play a role in determing the item's price as shown in the plot below.

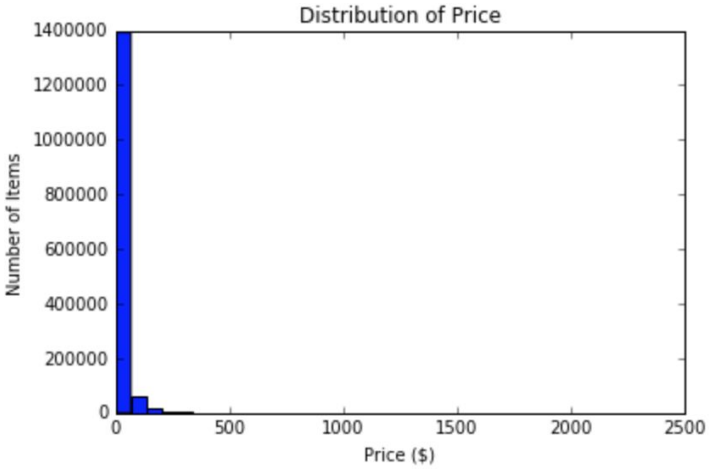




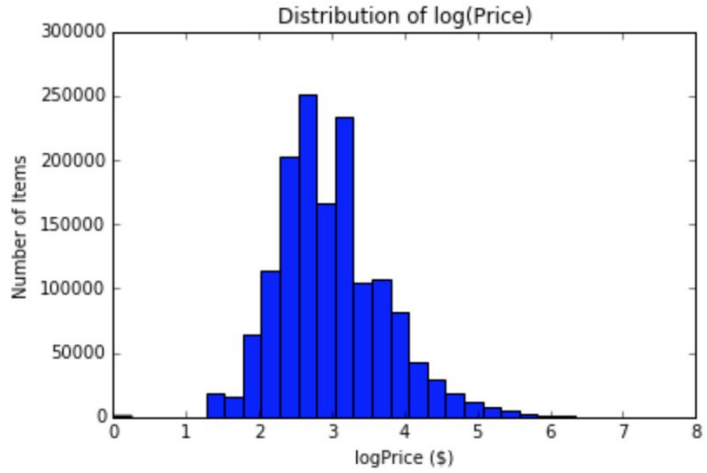
Analysis of Shipping:
The number of items that the shipping fee are paid by sellers is more than (but not by a lot) the number of items that the shipping fee are paid by buyers. It's as expected that the price of items are higher when sellers pay the shipping fee as can be seen from the plot below.

Analysis of Price:

The distribution of price is not really clear as can be seen from the plot below. There is a whole pile of very cheap items at one end, and the few expensive values aren't visible at all. There are 874 items that have $0 as the price and the price 99th percentile is $170.

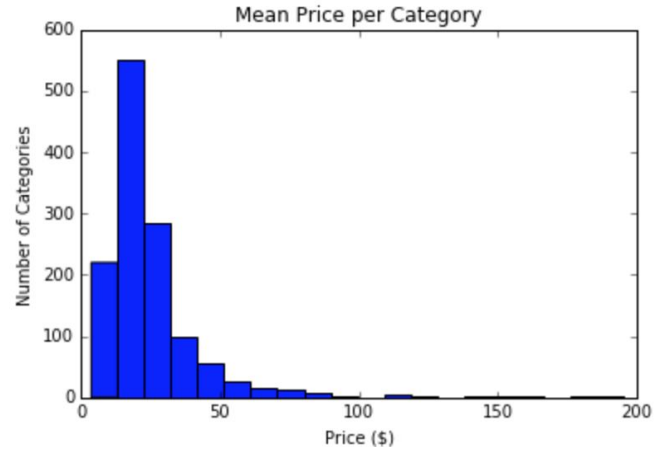


Looks like we should log transform the price in order to make the variable better fit the assumptions of underlying regression.

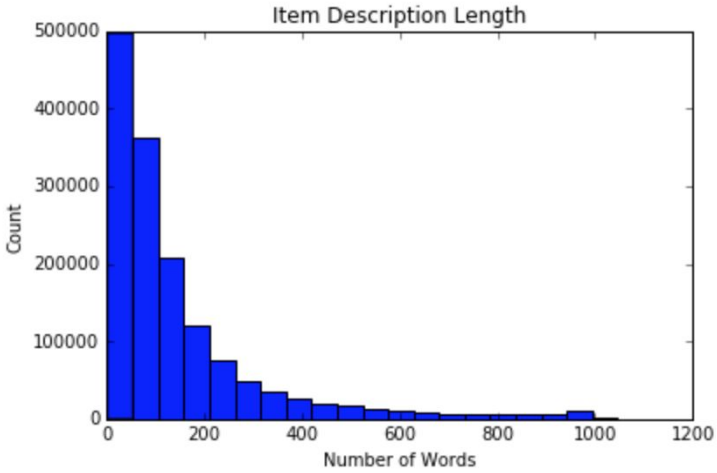


After log transforming price, we can see that the distribution of log(price) somehow follows a bell curve. Mostly, the one with $0 price are Women and Beauty categories as can be seen from the plot below.

Almost all categories have an average item price of <$100. With most of the categories having an average item price of <$50.

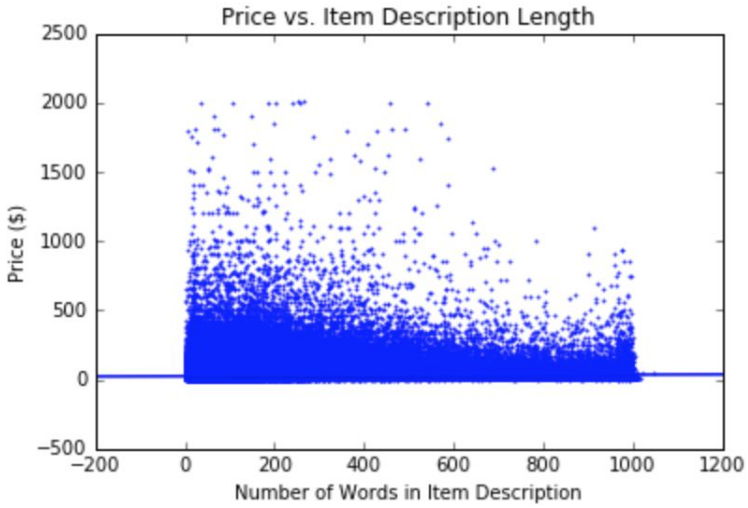


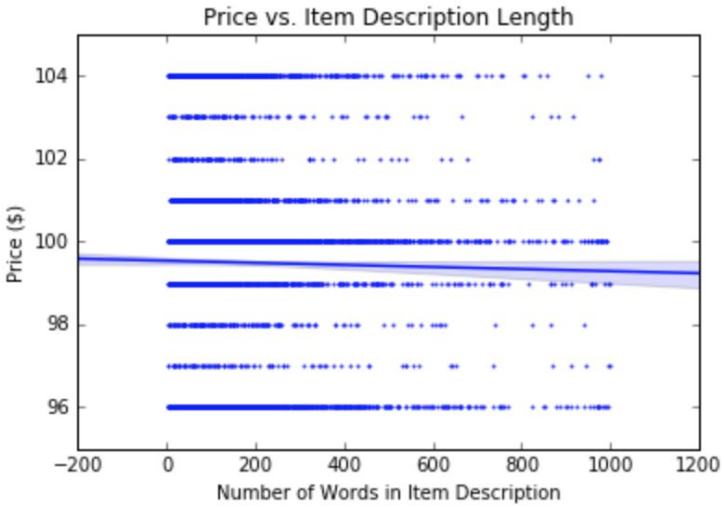
Analysis of Item Description:
The plot below shows the distribution of number of words in an item description.



The word cloud below shows what words are most common in the item description. They are standard words used when writing an item description.

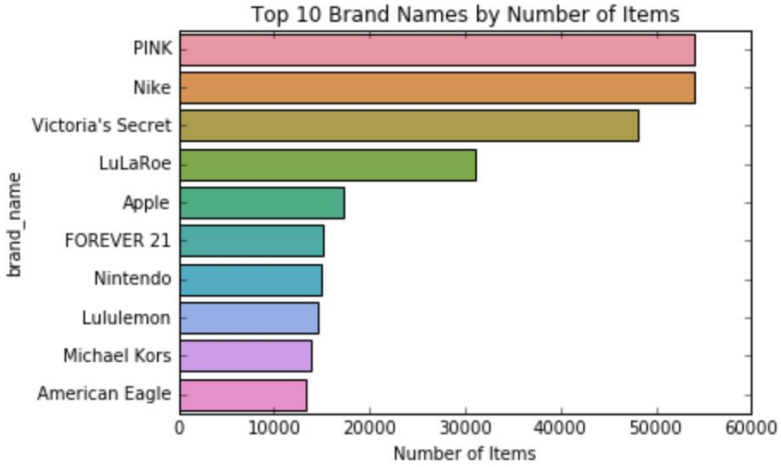
From the scatter plots below, we can see that there is no clear relationship between item description length and price.

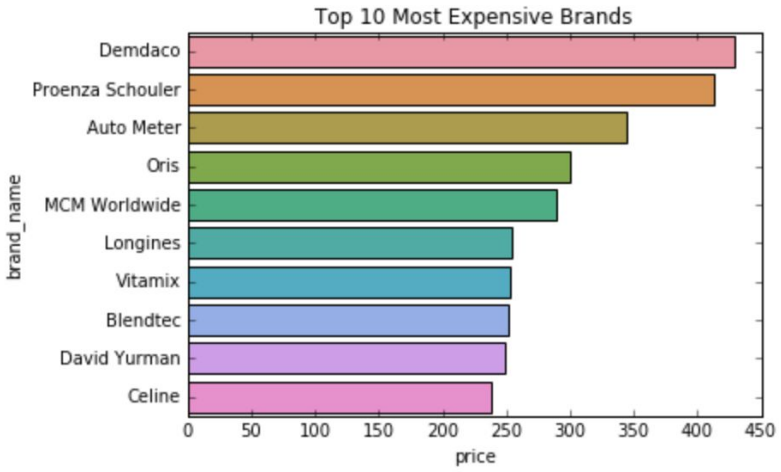




Analysis of Brands:

There are 4810 unique brands. The most expensive brands are luxury handbags, watches, and jewelry brands. PINK, Nike, and Victoria's Secret are the top 3 brands with most number of items.





Features Engineering

To produce the final data frame that we can feed to the machine learning model, we need to create features/extract features from different level of aggregations. The total number of features are 65. They are explained below.

- item_condition_id: The condition of the item
- shipping: Whether or not buyer pays the shipping fee
- brand_yesno: Whether or not the the seller specify the brand name
- category_yesno: Whether or not the seller specify the category
- item_desc_yesno: Whether or not the seller specify item description
- item_desc_len: Number of words in item description
- tfidf: The average tfidf for words in the item description
- *_price_category_name: Max/Min/Mean/Median price for item in a specific category name
- *_price_category_1: Max/Min/Mean/Median price for item in a specific category 1
- *_price_category_2: Max/Min/Mean/Median price for item in a specific category 2
- *_price_category_1_2: Max/Min/Mean/Median price for item in a specific category 1_2
- *_price_category_3: Max/Min/Mean/Median price for item in a specific category 3
- *_price_cond_category_name: Max/Min/Mean/Median price for item in a specific category name with a specific item condition
- *_price_cond_category_1: Max/Min/Mean/Median price for item in a specific category 1 with a specific item condition
- *_price_cond_category_2: Max/Min/Mean/Median price for item in a specific category 2 with a specific item condition
- *_price_cond_category_1_2: Max/Min/Mean/Median price for item in a specific category 1_2 with a specific item condition
- *_price_cond_category_3: Max/Min/Mean/Median price for item in a specific category 3 with a specific item condition
- *_price_brand_category: Max/Min/Mean/Median price for item in a specific category with a specific brand name
- *_price_brand: Max/Min/Mean/Median price for item with a specific brand name
- *_price_brand_cond: Max/Min/Mean/Median price for item in a specific category with a specific item condition
- *_price_cond: Max/Min/Mean/Median price for item with a specific item condition
- brand_name_price: score(ranking) of brands by price
- brand_name_count: score(ranking) of brands by number of items for a specific brand.

Train - Test Split

Since kaggle does not provide the testing data, the original training data is split into new training data (80%) and new testing data (20%) so that we can fit the model using the new training data measure the performance of the machine learning model on the new testing data.

Light GBM

The model used in this problem is Light GBM. To understand Light GBM, we need to understand gradient boosting. There are two main components involved in gradient boosting: a loss function to be optimized (auc, log-loss, etc.) and a weak learner (decision tree). Gradient boosting is an additive model where trees are added one at a time and a gradient descent procedure is used to minimize the loss when adding trees. The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve final output of the model.

Light GBM is relatively new and is a type of gradient boosting like XGBoost but with much faster run time and comparable performance. Instead of splitting the tree depth wise (XGBoost), Light GBM splits the tree leaf wise which results in a faster execution time. Splitting the tree leaf wise may lead to increase in complexity (and overfitting) but this can be overcome by tuning the parameter max-depth.

The main parameters of Light GBM are:

- Number of leaves: Main parameter that controls the complexity of the model.
- Max depth: Used to limit the tree depth and prevent overfitting
- Learning rate: A high learning rate can lead to overfitting but a lower learning rate is slower.

Light GBM Model Implementation

Light Gradient Boosting (LGBMRegressor) is performed on the training data using grid search and 5-fold cross validation to tune the parameters of the model. The target variable that we want to predict is log(Price).

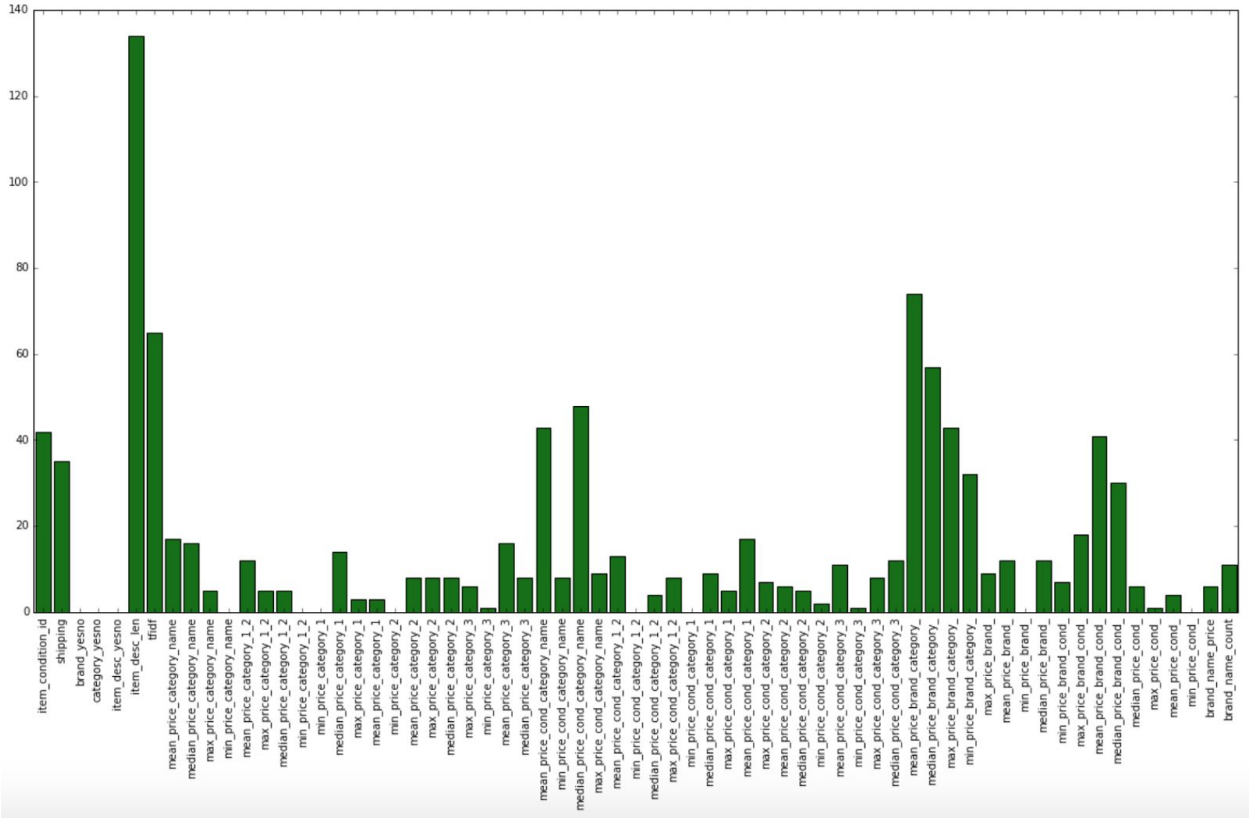
The parameters tuned are: learning rate, number of leaves, and max depth. The best parameters combination that produces the least error (RMSE) is:

- learning rate = 0.5
- number of leaves = 100
- max depth = 8

Using the best parameters combination, we refit the model and use the new model to predict log(Price) on the test data. The performance of the model is:

- mse on log(Price): 0.283866610945
- rmse on log(Price): 0.532791339029
- avg price error (average price difference): 11.604541683899463

The features importance plot is shown below.



After plotting the features importances, we can see that a few features are not important and the important features are only 25 (out of the original 65 features):

- Max_price_brand_cond_
- Median_price_brand_
- Mean_price_brand_category_
- Median_price_cond_category_name
- Mean_price_category_name
- Mean_price_brand_cond_
- Item_condition_id
- Shipping

- Mean_price_brand_
- Brand_name_count
- Mean_price_category_1_2
- Median_price_brand_category_
- Tfidf
- Mean_price_cond_category_3
- Max_price_brand_category_
- Item_desc_len
- Median_price_category_1
- Mean_price_category_3
- Median_price_cond_category_3
- Median_price_brand_cond_
- Mean_price_cond_category_1_2
- Median_price_category_name
- Min_price_brand_category_
- Mean_price_cond_category_name
- Mean_price_cond_category_1

Then, we retrain the model using only these 25 important features. The performance of the model on the test data is very similar to the performance of the model when using 65 features.

- mse on log(Price): 0.283881667401
- rmse on log(Price): 0.532805468629
- avg price error (average price difference): 11.603250468427452

Ridge Regression Model Implementation

Ridge Linear Least Squares Regression with CV is also performed as a comparison for the Light Gradient Boosting. The regularization coefficient alpha is also tuned to minimize error. The best alpha turns out to be 0.001. The performance of the model shown below is worse than the performance of Light Gradient Boosting.

- Best alpha: 0.001
- Mse on log(Price): 0.316908530359
- Rmse on log(Price): 0.562946294383
- avg price error (average price difference): 13.354043044080294

Conclusion

Light Gradient Boosting performs better than Ridge Regression to predict price in this problem.