



## Capstone Project Report

Machine Learning Engineer Nanodegree

Author: Xiao Ping Yang

June 2018

### I. Definition

After extensive research, I decided to run a Kaggle competition on Mercari Price Suggestion Challenge as my capstone project (<https://www.kaggle.com/c/mercari-price-suggestion-challenge>).

#### 1.1 Project Overview

Mercari (Latin word means “to trade.”, <http://www.mercari.com>) is a Japanese unicorn startup – call it an advanced version of Craigslist. It was founded in 2013 by Hiroshi Toshima, Ryo Ishizuka, Shintaro Yamada, and Tommy Tomishima. It is a community-sourced customer-to-customer mobile marketplace to securely buy and sell anything and everything. Mercari extended to the US and the UK in 2014.

It has raised \$116 Million since inception through five funding rounds and had 55 million downloads worldwide. The app witness more than 100,000 items updates per day. Mercari boosts its uniqueness by attaching an emotional value of things being trading on their app and help customers throughout the transaction.

To get one step further, in the age of AI and Data Sciences, Mercari recently launched this competition for \$100,000 in prize money. The competition is to predict the product’s selling price so they can guide the customers who are placing the products to sell (thus increasing their chances of selling the product for the right price and gaining the buyers’ trust of buying the product for the right price as well).

However it can be hard to know how much something used is really worth. Small details can mean big differences in pricing. For example, one of these sweaters cost \$335 and the other cost \$9.99. Can you guess which one’s which only based on item description?

Product pricing gets even 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.

In this competition, Mercari is challenging us to build an algorithm that automatically suggests the right price on a product for its customers.

#### 1.2 Problem Statement

Mercari challenges us to predict the price the item can be sold, so that they can make recommendation to the users when the products are listed for sell.

The training dataset (with price value) contains about 1.5 million observations. And the given datasets have close to 700,000 observations in the test file, so I am asked to predict the price value for each item in this dataset as my submission.

The tab-delimited training files contain eight columns – ID, Name, Item Condition, Category, Brand Name, Price, Shipping and Item Description.

Out of given eight variables, four columns contain numeric data (ID, Price, condition, and Shipping) and another four columns contain string data (Name, Category, Brand and Item Description). The competition can be done to perform Natural Language Processing on given descriptions to find out which words and descriptions correlated with the product price. Brand name and categories should have their own and very important weights as well.

As the price to predict is continuous numeric value, this is a supervised machine-learning problem. Regression models might be good solutions to solve the problem.

This competition is kernel-only competition, which means I have to upload my script into Kaggle and run the code at their cloud machine, and ensembles have to be done in a single kernel.

System specification: 16GB ram, 1 GB disk, ~4 cores, only 60 minutes to train and predict!

Considering the limitation of kernel, the training speed is the major factor to select models. I would consider Ridge Regression auto model as a bench model, and then compare it with the advanced models such as Ridge Sag model and LightGBM Model. And then tune parameters on the selected model for submission.

### 1.3 Metrics

The evaluation metric for this competition is Root Mean Squared Logarithmic Error. The RMSLE is calculated as below:

$$e = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

e is the RMSLE value (score)

n is the number of samples

$p_i$  is the prediction of the price

$a_i$  is the actual sale price

$\log(x)$  is the natural logarithm of x.

## II. Analysis

### 2.1 Data Exploration

The training file has 1,482,535 records and the testing file has 693,359 records.

- Training file

The training file contains 8 features.

Feature Names	Description	Data Type
train_id	id of the listing	Int64
name	title of the listing	Object
item_condition_id	condition of the items provided by the seller	Int64
category_name	Category of the listing	Object
price	Price that the item was sold for	Float64
shipping	1 if shipping fee is paid by seller and 0 by buyer	Int64
Item_description	Full description of the item. Price information in the description is removed	Object

First 10 rows of training file as below:

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 Décor/Home Décor 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
5	5	Bundled items requested for Ruie	3	Women/Other/Other	NaN	59.0	0	Banana republic bottoms, Candies skirt with ma...
6	6	Acacia pacific tides santorini top	3	Women/Swimwear/Two-Piece	Acacia Swimwear	64.0	0	Size small but straps slightly shortened to fi...
7	7	Girls cheer and tumbling bundle of 7	3	Sports & Outdoors/Apparel/Girls	Soffe	6.0	1	You get three pairs of Sophie cheer shorts siz...
8	8	Girls Nike Pro shorts	3	Sports & Outdoors/Apparel/Girls	Nike	19.0	0	Girls Size small Plus green. Three shorts total.
9	9	Porcelain clown doll checker pants VTG	3	Vintage & Collectibles/Collectibles/Doll	NaN	8.0	0	I realized his pants are on backwards after th...

There are about 1.5 million records provided in the training file. The average selling price is \$26.74 with standard deviation of \$38.59. The description of the training file as below:

	train_id	item_condition_id	price	shipping
count	1482535.00	1482535.00	1482535.00	1482535.00
mean	741267.00	1.91	26.74	0.45
std	427971.14	0.90	38.59	0.50
min	0.00	1.00	0.00	0.00
25%	370633.50	1.00	10.00	0.00
50%	741267.00	2.00	17.00	0.00
75%	1111900.50	3.00	29.00	1.00
max	1482534.00	5.00	2009.00	1.00

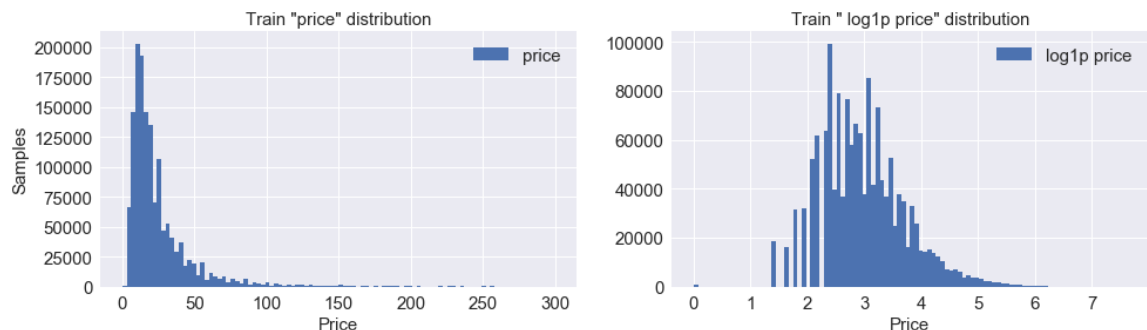
- Testing file

Other than the price columns, the testing file contains the same columns. That makes sense, as I am required to predict the price in the testing file.

## 2.2 Data Exploratory Visualization

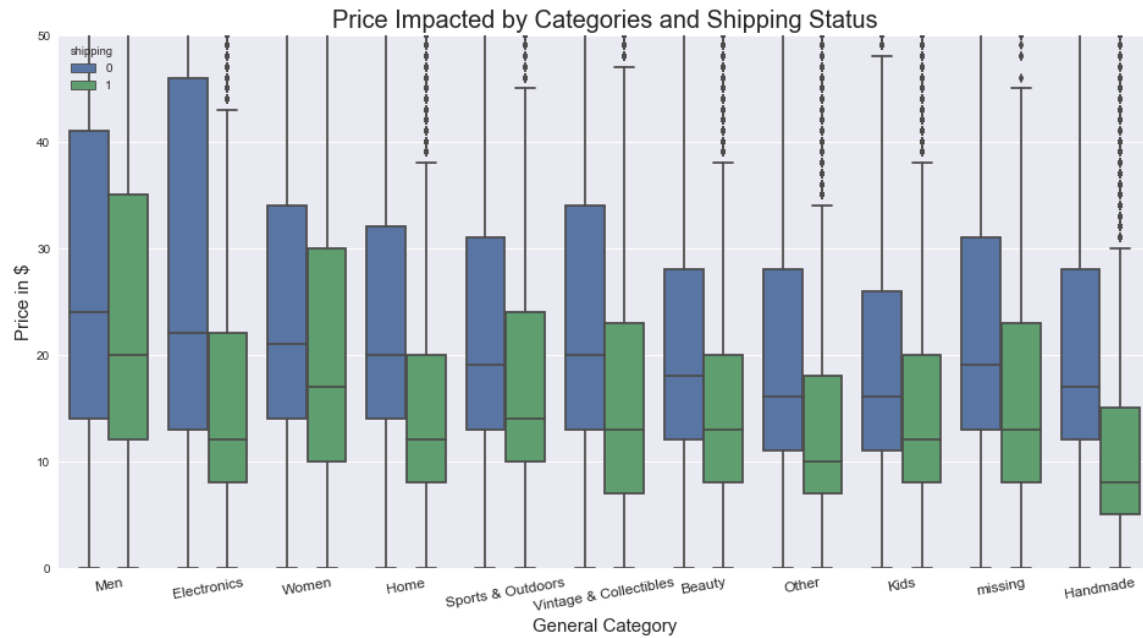
- Target variable: price

The target variable price ranges from \$0 to \$2009. Let's look at the histogram of prices.

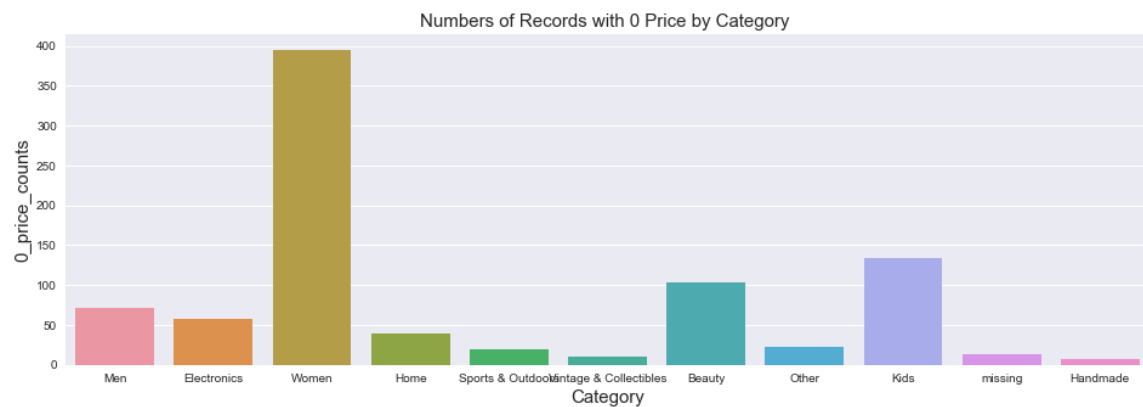


The distribution of the price variable is heavily skewed to the left. The  $(\log \text{price} + 1)$  appears to be centered around 3 and has a longer right tail due to the 0 bound on the left.

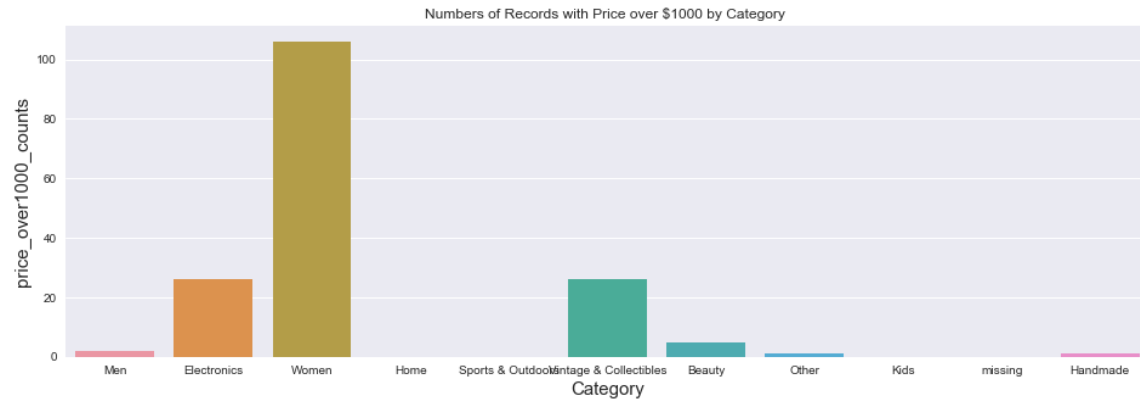
And the target variable price is heavily impacted by category and shipping status. Men category has the highest average price and the handmade category has the lowest average price. The boxplot between price and categories by shipping status as below:



Within the training file, there are 874 items which were gave away free. Among the first level categories, women, kids and beauty are the top three categories have the most free items. The plot of 0-price-counts by different category as below:



There are 167 items were sold more than \$1000. The category of women has more than 100 records sold over \$1000 dollars; electronics and sports & outdoor collections have more 20 items sold over \$1000 respectively.

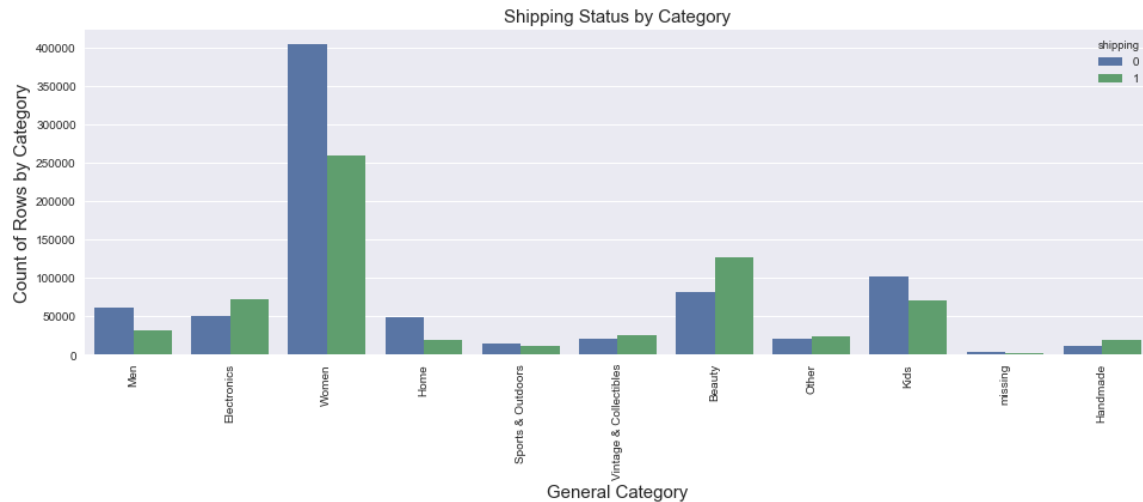


As expected, the newer items can be sold higher prices. As the plot shows below, some newest items can be sold over \$2000, but the item condition with value equals 5 (means condition is bad) is hardly can be sold over \$500. The price distribution by item condition is as below:



#### ▪ Input feature: shipping

Among eight input features, women, men, home and kids categories have more shipping items than non-shipping items. Meanwhile, electronics and beauty categories have more non-shipping items. The rest categories have roughly equal shipping and non-shipping items. The plot of shipping status by category is as below:



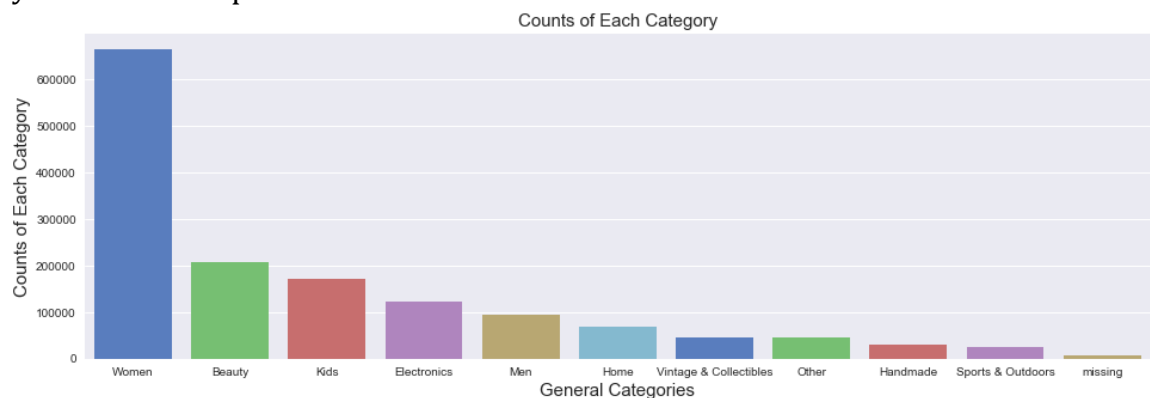
- **Input feature: item\_condition\_id**

There are 5 values in item\_condition\_id. Value 1 means the item is the newest, and 5 means the item is least good condition. Item condition counts plot by different category as below:

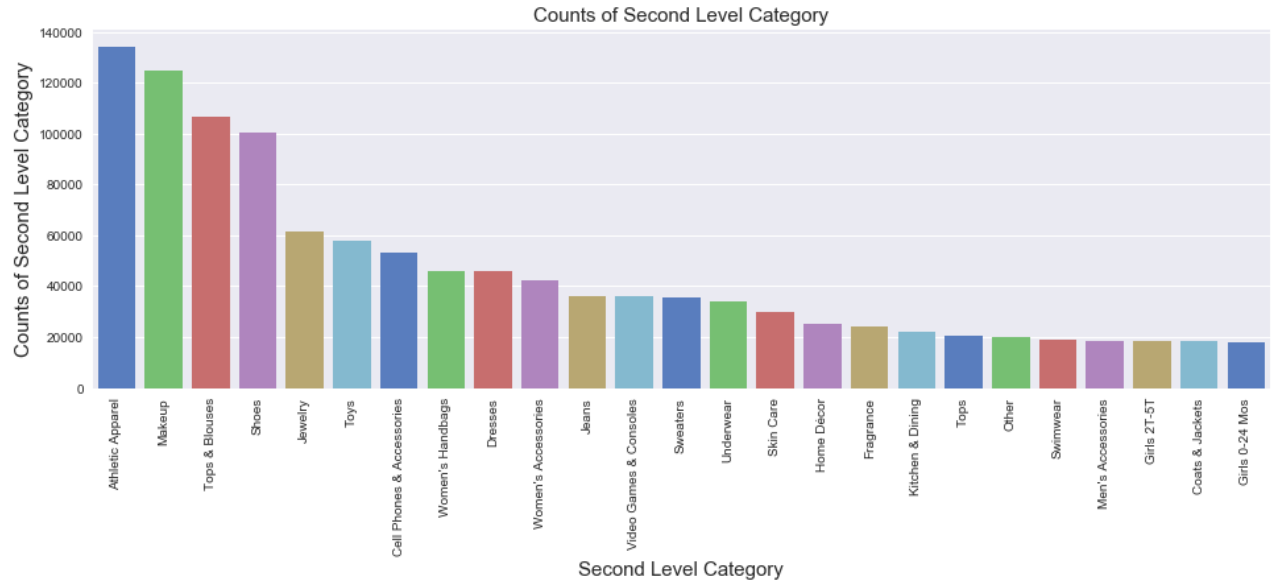


- **Input feature: category**

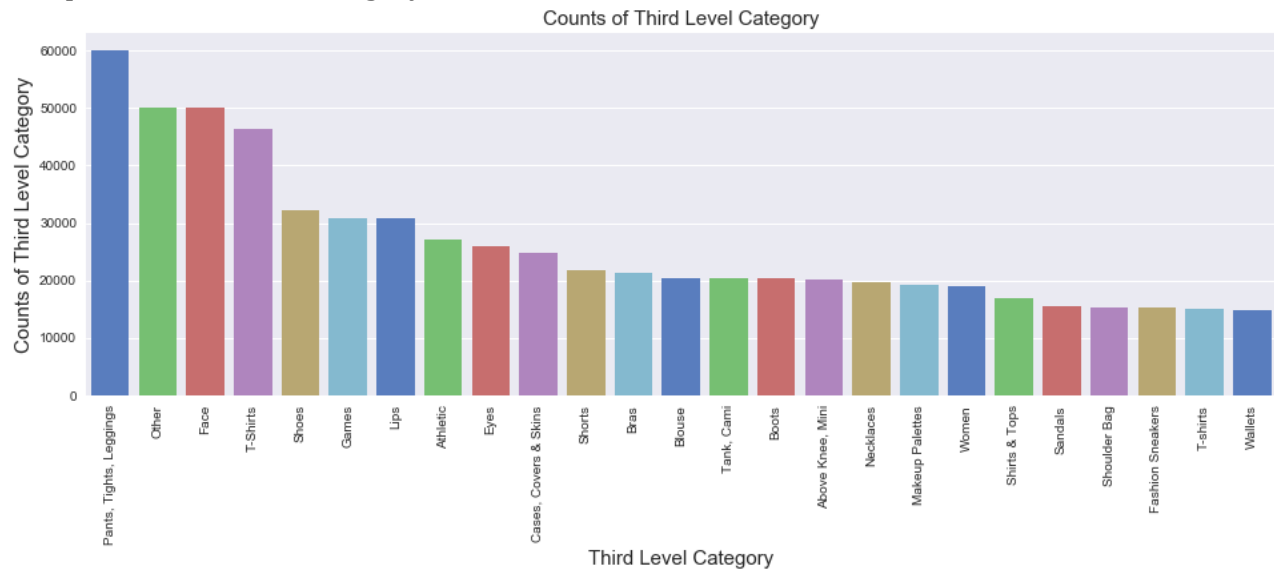
For the category column, the value can be split by '\ ' into 3 levels. There are 9 values in the first level category: women, beauty, kids, electronics, men, home, vintage&collections, handmade,sports&outdoors, other and missing. The first level category records count plotted as below:



Among the second level category, the plot of records count by sub-category as below:



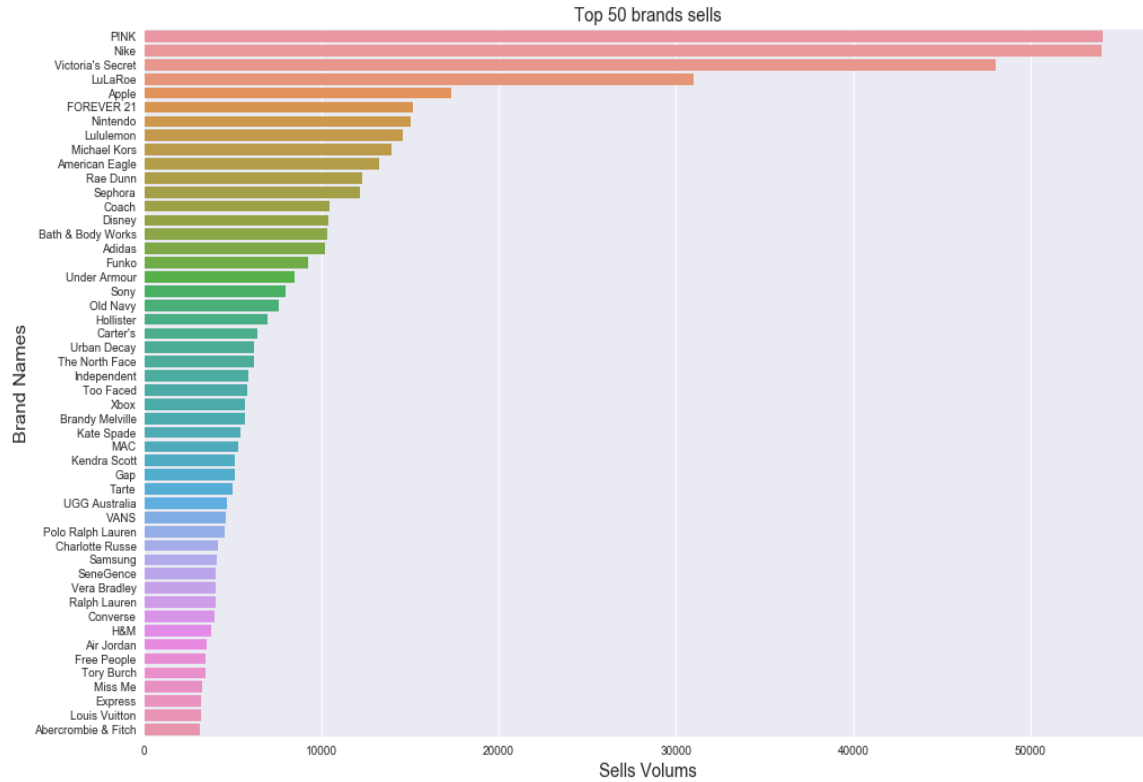
The plot of third level category counts as below:



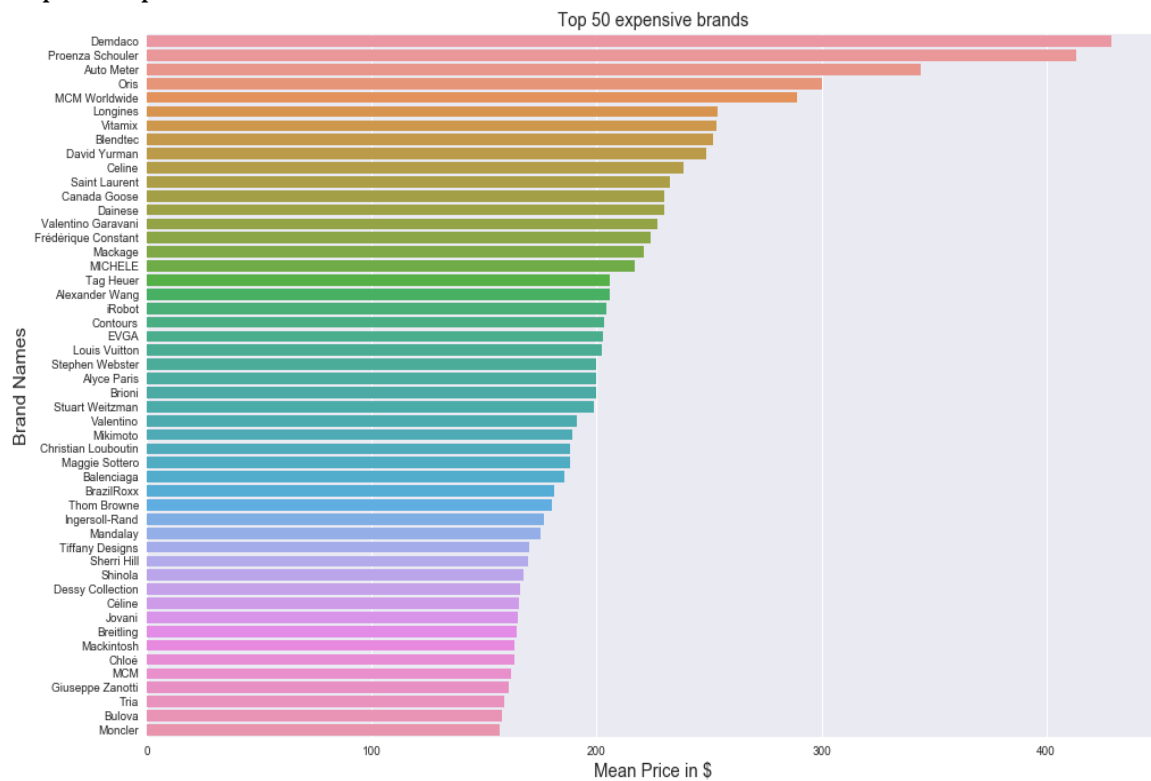
- **Input feature: Brand\_Name**

There are total 4810 different brands in the training file, among them, 3566 brands appeared more than once. Top 50 brands sold as below:

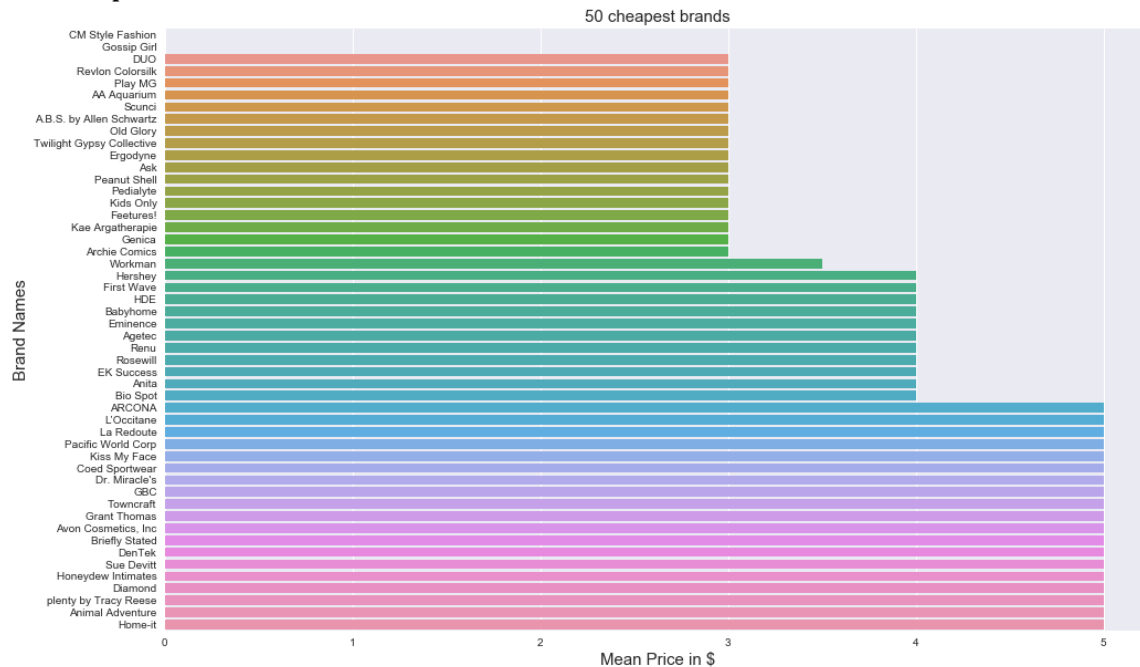




Top 50 expensive brands as below:



50 cheapest brands as below:



- **Input Feature: Name and Item\_description**

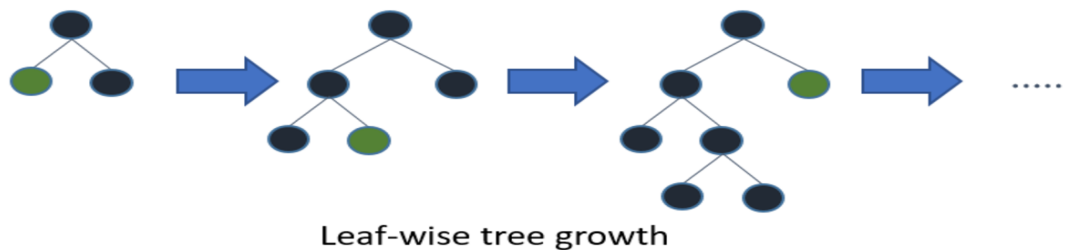
Name and Item\_description are texts fields. The longest name field has 17 words and the longest item\_description field has 245 words.

Wordcloud is a visual representation of text data. It displays a list of words, the importance of each being shown with font size or color. It is useful for quickly perceiving the most prominent terms.

The wordcloud for name field as below:







When growing the same leaf, leaf-wise algorithm can reduce more loss than a level-wise algorithm.

### **Why lightGBM is a good fit for this competition?**

Light GBM is prefixed as 'light' because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. For this challenge, the Kaggle Kernel only offers 16GB RAM and 60 minutes running time, lightGBM definitely would help to achieve better ranking.

Another reason of why LightGBM is fit for this competition is it focuses on accuracy of result thus data scientists are widely using LGBM on Kaggle to work on Mercari challenge.

LightGBM is not advisable to be used on small datasets. LightGBM is sensitive to overfitting and can easily overfit small data. Normally the datasets should have more than 10,000+ rows. For Mercari competition, the training file has about 1.5 million rows and the testing file has about 700,000 rows so that the overfitting would not be an issue to apply this algorithms.

In addition, the implementation of LightGBM is easy, the only complicated thing is parameter tuning. LightGBM covers more than 100 parameters but using only basic parameters work very well.

## **2.4 Benchmark**

I initially planned to use Random Forest as benchmark model. Unfortunately after the data preprocessing, the input data size is so large that it took more 12 hours for Random Forest Model to finish the training. As the competition only allows one hour to run in the kernel that I have to give up this model as the benchmark model.

Ridge Regression has 2 solvers, one is 'auto' and another one is 'sag'. 'auto' chooses the solver automatically based on the type of data. 'sag' uses a Stochastic Average Gradient descent. Both of solvers support dense and sparse data.

I would choose the following model as a benchmark model:

Model Name: Ridge Regression

Solver: 'auto'

Evaluation Metrics: RMSLE

### III. Methodology

#### 3.1 Data Preprocessing

- Filling the missing data

Fields	Filling values
brand_name	'unknown'
item_description	'None'
shipping	0
category_name	'None/None/None'

- Handling the brand name  
There are total unique 4810 brands, but only 3566 brands appeared more than once. Assign 'other' value to the brand only existing once.
- Cleaning up the name and item\_description fields  
Convert all of the words to lower case  
Only keep words and numbers, and remove all of the punctuation
- shipping  
Applying one hot encoder to fit and transform
- Generate a new feature 'object'  
According to the name feature, the last 2 words most likely describe what the item it is. If extract the item name without the adjective words, it would have concise information helping to know what exactly what product is.

#### 3.2 NLTK Preprocessing

Fields	Methodologies	Descriptions
name	LemmaVectorizer	<ul style="list-style-type: none"> <li>Remove stop words</li> <li>Convert the text documents to a matrix of token counts using count vectorizer</li> <li>Using stemming to reduce inflectional forms and derivationally related forms of a word to a common base form</li> </ul>
brand_name	LemmaVectorizer	
category_name	nltk countvectorizer	<ul style="list-style-type: none"> <li>Convert a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts using <code>scipy.sparse.csr_matrix</code>.</li> </ul>
item_description	nltk countvectorizer	
object	nltk countvectorizer	

### 3.3 Training Models

Training Models	Key Parameters	Parameters Description
Ridge Regression Model	<code>solver='auto'</code>	Choose the solver automatically based on the type of data.
Advanced Ridge Model	<code>solver='sag'</code>	Uses a Stochastic Average Gradient Descent
	<code>alpha</code>	regularization strength, improves the condition of the problem and reduces the variance of the estimates. Must be a positive float. Larger values specify strong regularization
	<code>max_iter</code>	max number of iterations for conjugate gradient solver
	<code>tol</code>	precision of the solution
	<code>fit_intercept</code>	whether to calculate the intercept for this model
	<code>normalize</code>	if true, the regressors X will be normalized before regression by subtracting the mean and dividing by the l2-norm.
LightGBM Model	<code>learning rate</code>	The learning parameter control the magnitude of the change in the estimates
	<code>max_bin</code>	Max number of bins the feature value will bucket in
	<code>num_leaves</code>	Number of leaves in full tree
	<code>min_data_in_leaf</code>	Minimum number of records a leaf may have
	<code>bagging_freq</code>	The fraction of data to be used for each iteration
	<code>max_depth</code>	It describes the maximum depth of tree
	<code>bagging_fraction</code>	Fraction of data to be used for each iteration and generally used to speed up training

	lambda_l1	L1 regularization, ranges from 0 to 1
	lambda_l2	L2 regularization
	application	'regression' for regression; 'binary' for binary classification 'multiclass' for multiclass classification problem
	early_stopping_round	Model will stop training if one metric of validation data doesn't improve in last early stopping rounds. This will reduce excessive rounds.

### 3.4 Implementation

The implementation of my solution is contained in the attached *Capstone\_report.ipynb* file.

The file contains the whole pipeline including the following:

- Data Exploration
  1. Using pandas loading the data in memory.
  2. Using matplotlib.pyplot to visualize data on price distribution impacted by categories, shipping status and item conditions. And view top 50 expensive brands sold.
  3. Using wordcloud library to display text fields
- Data Preprocessing
  1. cleanName(text) function to remove punctuations and unrecognizable characters
  2. getLastTwo(text) function to create a new feature which most likely can identify what item is.
  3. create a custom function LemmaVectorizer(CountVectorizer) to parse brand and name features.
  4. CountVectorizer function is used to parse category names and item description
- Benchmark model training  
Model name: Ridge  
Parameters: Solver= 'auto'
- Advanced model(I) training  
Model name: Ridge  
Parameters:

Parameters	Values
alpha	0.29
Max_iter	800

solver	'sag'
tol	0.001
Fit_intercept	True

- Advanced model(II) training  
Model name: lightGBM  
Parameters:

Parameters	Values
Max_bin	255
Min_data_in_leaf	1
Learning_rate	0.15
application	'regression'
Max_depth	20
Num_leaves	90
Bagging_freq	0
Bagging_fraction	0.5
Feature_fracture	1
Lambda_l1	2
Lambda_l2	0
bin_construct_sample_cnt	50000

### 3.5 Refinement

After the trainings of base model and compared with the advanced models, lightGBM produced way better score so that I choose to tune lightGBM model for the most optimized score.

There are two common methods of parameter tuning: grid search and random search. Each has their pros and cons. Grid search is slow but effective at searching the whole search space, while random search is fast, but could miss important points in the search space. Luckily, a third option exists: Bayesian optimization.

After extensive research, I choose a Python module caller 'hyperopt' to implement of Bayesian optiomization.

The search spaces as below:

Parameter name	method	Value
Learning_rate	uniform	(0.05,0.25)
Max_bin	choice	(255,510,765)
Max_depth	choice	[-1,5,10,15,20,25]
Num_leaves	choice	[60,90,120]



Bagging_frequency	choice	[0,5,10]
Bagging_fraction	uniform	(0.25,0.6)
Lambda_l1	uniform	(0,2)
Bin_construct_sample_cnt	choice	[20000,50000,70000]

And the tuned best parameters :

Parameter name	Best Value
Learning_rate	0.18
Max_bin	765
Max_depth	25
Num_leaves	120
Bagging_frequency	5
Bagging_fraction	0.43
Lambda_l1	1.1
Bin_construct_sample_cnt	70000





## IV. Results

The training file was split into train and validation datasets with 15% of datasets for validation . After the models are trained, I use the test file to predict the target value and generate the output file , and then uploaded the submission file into Kaggle. Kaggle ranked my submission based on RMSE.


All of the model results as below:

Model Name	Validation RMSE	Kaggle Public RMSE	Kaggle Private RMSE
Ridge Auto Model	0.4733	\	\
Ridge Sag Model	0.4734	0.4471	\
LightGBM Model	0.4138	0.41987	\
Tuned LightGBM	0.4111	0.41945	0.42048

The final model as implemented in the refinement section gave a big improvement on the leaderboard, improving to 0.41945 RMSE from 0.41987 in the initial lightGBM model. This gave me **97<sup>th</sup> out of 2384** teams on the final ranking.


95	▲ 7	newbies		0.42048	8	7mo
96	▲ 350	shiMu		0.42048	9	7mo
97	▲ 234	Electric_Brain		0.42048	67	7mo
98	▲ 134	Tilted Axis		0.42048	19	7mo
99	▲ 239	Vroem Vroem		0.42048	47	7mo

This competition awarded me with a Silver Award. Combined with the another silver award received on a previous competition, I get promoted to Kaggle Competition Expert, which is a great achievement for my learning process.



**xiao**  
 ML Engineer at YY  
 Toronto, Ontario, Canada  
 Joined 2 years ago · last seen in the past day

Followers 3  
 Following 28

  
**Competitions Expert**

[Home](#)
[Competitions \(6\)](#)
[Kernels \(3\)](#)
[Discussion \(52\)](#)
[Datasets](#)
...

[Edit Profile](#)

Competitions Expert	Kernels Contributor	Discussion Contributor
<div> <div>Current Rank</div> <div><b>1901</b></div> <div>of 91,096</div> </div> <div> <div>Highest Rank</div> <div><b>1571</b></div> </div>	<div>Unranked</div>	<div>Unranked</div>
<div> <div>🥇</div> <div>0</div> </div> <div> <div>🥈</div> <div>2</div> </div> <div> <div>🥉</div> <div>0</div> </div>	<div> <div>🥇</div> <div>0</div> </div> <div> <div>🥈</div> <div>0</div> </div> <div> <div>🥉</div> <div>0</div> </div>	<div> <div>🥇</div> <div>0</div> </div> <div> <div>🥈</div> <div>0</div> </div> <div> <div>🥉</div> <div>15</div> </div>
<div> <a href="#">Web Traffic Time Series Fo...</a>  <small>🏆 · 10 months ago · Top 5%</small> </div> <div> <a href="#">Mercari Price Suggestion C...</a>  <small>🏆 · 7 months ago · Top 5%</small> </div>	<div> <a href="#">TF Learn Approach for Digi...</a>  <small>a year ago</small> </div> <div> <a href="#">A Simple TensorFlow Conv...</a>  <small>a year ago</small> </div>	<div> <a href="#">ELI5 for Mercari</a>  <small>🏆 · 9 months ago</small> </div> <div> <a href="#">Official external data thread</a>  <small>🏆 · 8 months ago</small> </div>
<div>48<sup>th</sup> of 1095</div> <div>97<sup>th</sup> of 2384</div>	<div>1 vote</div> <div>0 votes</div>	<div>4 votes</div> <div>2 votes</div>

## V. Conclusion

My final solution greatly outperforms the benchmark solution, going from a 0.4733 RMSE score down to 0.41945. I feel the final solution presented here has been able to well capture the signal from the data in a way significant to produce an useful model.

I think my final solution does fit what I expected to have for this problem. It would be an appropriate model to use in actual price prediction situation for real business production.

### 5.1 Improvement

The first place solution for this competition achieved 0.3875RMSE. Compared with their solution, I think I can have the following areas to improve:

- Improve feature preprocessing
  - Name field: Using n-grams from name features can significantly improve the score
  - Text concatenation

My solution extracted the last 2 words from item description as a new feature. However the winning solution concatenated all of the text features {name, item\_description,category,brand} for a new single field. Per their presentation, this was the key reason for the score down to 0.4 RMSE.

- Models

The winning team stacked multiple models of Keras deep neural networks, 3x Convolutions1D neural networks and keras sparse MLP .

Deep learning is the area I just get started but not very confident about . I plan to spend more time to polish my deep learning skills and knowledge.

- Multi-core processing

The winning solution trained 4 models on 4 cores. One model on 1 core. As I didn't know how to set up multiple cores processing, I only run 1 model on 4 cores that reduced my opportunity for better optimization.

This is the area I should learn and improve as well!