……………………………………………...……………………………………………...……………………………………………...……………………………………………………………...………………

**Name:** Akarshan kumar

**Email address:** akarshan.1711@gmail.com

**Contact number:** 7903299199

**Anydesk address: 114 999 665**

**Years of Work Experience: 0**

**Date: 16th Nov 2020**

……………………………………………...……………………………………………...……………………………………………...……………………………………………………………...………………

**Self Case Study -1:** Mercari Price Suggestion Challenge

# **Overview**

Mercari Price Suggestion Challenge is a competition hosted on Kaggle. This Japanese community empowered firm Mercari provides a shopping app that enables sellers to list their items on the site for potential buyers. This is a customer to customer model for trade. Mercari wants and tries to help the sellers on its platform by suggesting prices for their goods while the enlisting of goods takes place in real time with the data provided by the customer beforehand about the items. This is a very daunting task as sellers are enabled to put anything i.e. single item, antiques, exquisite ones, bundled or set of items, used ones, broken, tampered, brand-new, etc kind of items on the site with what we suppose some relevant information about the item. The site has to account for the range of goods that can be sold, the specification/condition and price trade-offs, and many niche e-commerce trends. The site has images which are not part of the competition, whereas we can use the name or title, item condition ranging 1 to 5 from being good to worst, a category it belongs to, brand name, shipping with value 1 for free shipping or 0 for charges apply for shipping and item description with short crisp to lengthy paragraphs to make the product more appealing. Lastly, there is the Price tag, the target value against which our regression model’s predictions have to be compared using root mean square log error.

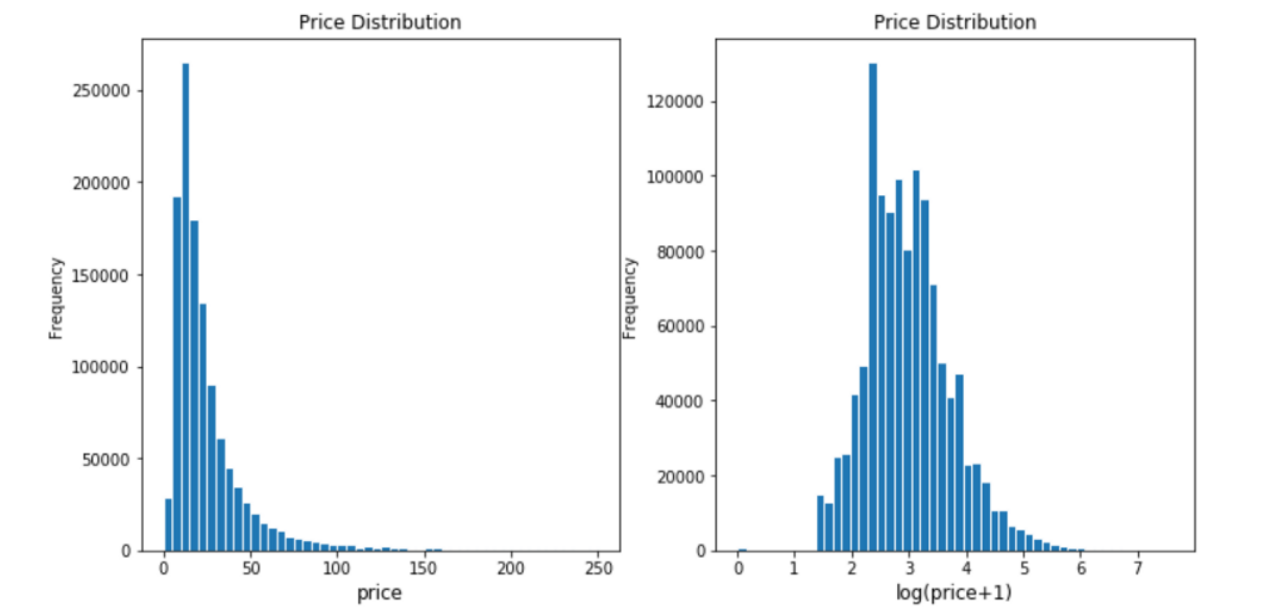
There is one more dimension to this challenge other than a normal kaggle challenge as this one has a fixed kernel capacity of 16 gb ram and 1 gb of hard disk space with only four cores to work with. On top of that the code must get executed in under 60 minutes. The first stage of challenge had comparable amounts of train and test data. The next stage involved a new test set which is internally accessed by kaggel that is 5 times bigger than any of yet the provided datasets, so optimization is the next challenge where code should not go default on memory with that huge amount of data being processed. This competition builds on the 2 aspects of the problem one being the Data-Science approach , mixed with the code optimization techniques.

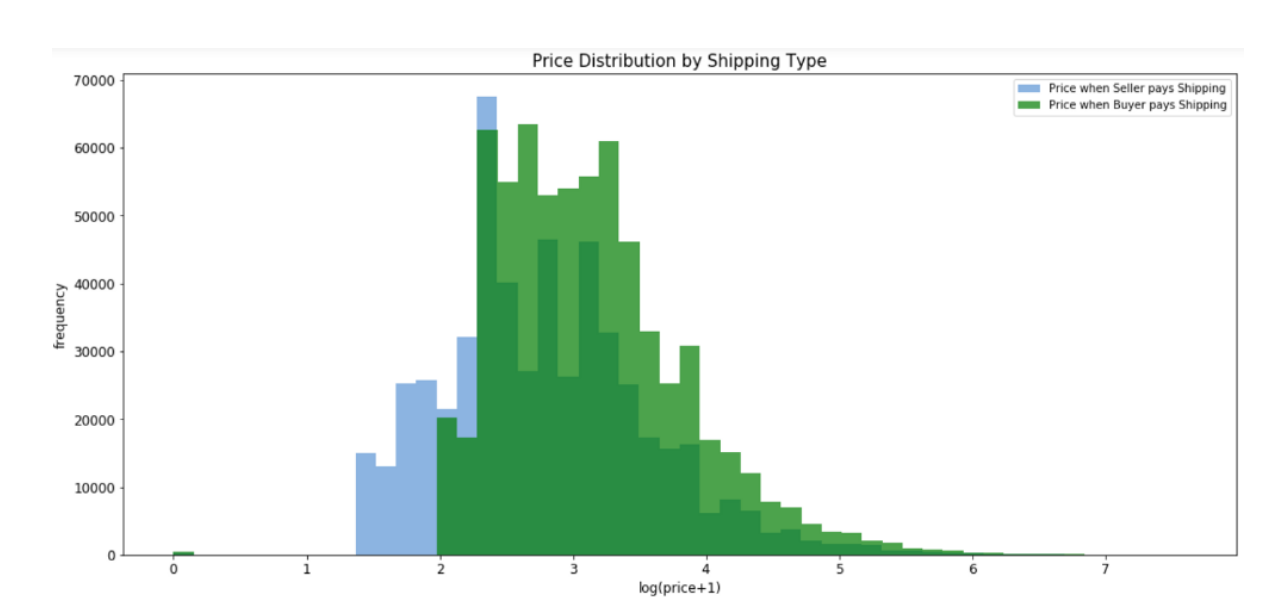
**Research-Papers/Solutions/Architectures/Kernels**

1. [hion-with-python-64531e64186dttps://towardsdatascience.com/machine-learning-for-retail-price-suggest](https://towardsdatascience.com/machine-learning-for-retail-price-suggestion-with-python-64531e64186d)

Machine learning for retail price suggestion blog

* This blog shows a vanilla approach to the problem and yet gets a reasonable ‘rmse’ score of 0.462 on log transformed price.
* This shows that we cannot discount the feature extraction power of simple predictor models in this problem.
* The blog starts with simple EDA on the dataset. The price distribution is shown to belong to the log distribution. A simple logarithmic transform of price is gaussian in nature.



* It is also revealed that the avg price of items surge considerably if the shipping charges exist i.e. the buyer pays for the shipping of items, which is an obvious discovery.
* Category and price trade off seems non-existent as all kinds of prices are asked for every kind of category but minor upward median shifts in category 5.
* Missing value imputation includes dropping rows with no price, filling NA for category and brand name. And tf idf with mindf are used.
* As stated earlier this blog does not perpetrate any kind of feature extraction or engineering, which is impairing the data available but that does go on to show the raw power of LGBM.
* A gradient boosting model with base learners as trees, LGBM is used with proper preparation of data and hyper parameters tuned to get the final model for evaluation. The evaluation is also done with rmse metric with log transformed prices, described by the kaggle competition.

1. <https://www.kaggle.com/vrtjso/mercari-eda-more-info-than-you-can-imagine>

An EDA kernel from kaggle

* This kernel mainly focuses on the EDA of the mercari price suggestion data and does not have any predictive analysis.
* The first few graphs show the distribution of different features.
* Following graphs show the hierarchical clustering in the different levels of categories along with their distribution.
* The diagnosis is followed by the insights from brands columns alone and its interaction with the item name column.
* Final insights are drawn from the interaction of price features with item description length and brand value.

1. <https://www.kaggle.com/rumbok/ridge-lb-0-41944>

A Ridge regression kernel with great feature extraction

* This kernel shows that intuitive data science techniques still matter as a vanilla non-ensemble technique achieved a score of 0.41944 with excellent feature generation tricks.
* This kernel makes use of Damerau Levenshtein distance, which is the number of operations (consisting of insertions, deletions or substitutions of a single character, or transposition of two adjacent characters) required to change one string into another. This distance is used to create a dictionary of strings as keys and its nearest words with a default distance as 3 as the values. This dictionary is used for suggestions to find the best replacement of NAN values in brand features using the description and name of the item. The words form description and name of items were used as keys, then the best brand suggestions, if there exists within a distance of 3, from the corpus of values is suggested. This was a very well thought Data Science approach to impute the missing values in a feature.
* This function is part of a pipeline which includes regex to fix another kind of miss spellings, preprocessing and normalising features, getting rid of stop words, lemmatization and stemming of the dataset and splitting the data in train, cv and test and then vectorization to finally input to a simple Ridge model for prediction.

1. <https://www.kaggle.com/gspmoreira/cnn-glove-single-model-private-lb-0-41117-35th>

A Deep NLP approach to solve Mercari price suggestion challenge

* The kernel is trying to use the most out of textual facts present in the features of data, on top of it extract its features using CNN and Feed Forward models.
* There is an interesting find in this kernel in the EDA process, using the coefficient of variation. The item description for sub-category products with the highest coefficient of variation states that these are nothing but bundles of many products combined and hence this skewness is price is prevalent.
* One great example of missing value imputation is shown in the Brand feature. Wherever the Brand value is null, the kernel defines a heuristic to find it in item name description using all the known brand names and their corresponding common category values.
* Text vectorization included TF-IDF vectorization but with bi grams using hashing vectorization to keep memory constraints intact, then feed this sparse matrix in the embedding layer for its usage in the neural network .
* A CNN is used instead of RNNs because of the kernel constraint in the competition, and a well thought architecture is made where the textual data is fed first through CNN and Avg pooling and a strong representation is formed. Then these are parallely fed to the deeper layers of architecture with numerical data(skip connections were used to generate this architecture).
* Textual descriptions were limited to 20 words and shorter descriptions were padded. Less frequent words were replaced with special tokens.
* At last with normalization and dropout the deep learning model is trained within 48 minutes and a score of 0.41117 was obtained.

1. <https://www.kaggle.com/c/mercari-price-suggestion-challenge/discussion/50256>

or

<https://www.youtube.com/watch?v=QFR0IHbzA30&t=1276s>

The mercari price suggestion challenge’s winners’ solution

* This kernel is a 2 man work by Pawel Jankiewicz and Konstantin Lopuhin.
* They explain that RMSLE is nothing new but the same RMSE error with log transformed inputs. This was so because the target column price belonged to log distribution.
* Because of the optimization issues with this challenge, the winners’ had a modular approach, i.e. the python code was split into modules and a self running script was used. A pipeline based approach was prominent as memory constraints were high.
* Their analysis states that most feature generation with the given data yielded futile results. Simple stemming, BOW with/without TF-IDF and one hot encodings were used with a Feed Forward network as this was more efficient in terms of time than the CNN or RNN ones.
* A simple tri-gram feature worked.
* A reverse engineering approach was prevalent to debug the model. Where the models were failing to predict well, were great cases to enhance the algorithm.
* A good approach to getting diversification was not to ensemble different models but the same model with different datasets(splitting datasets). They also used the bagging trick by averaging weights of models which were overfitted( high variance) with an extra epoch while training with huber loss, as it penalises the outliers less and Adam optimizers.
* They created 64 bins in which price could belong and this means that it was a classification then regression approach which yielded a score of 0.37758

# 

# **First Cut Approach**

1. Starting off with EDA, a correlation shall be defined, if there is any, between the brands and their asking price. If there seems to be a correlation then binning of the brands will be done as a new feature with an appropriate number of buckets of price.
2. A same approach can be described for a combination of Brands with Condition they are enlisted so to watch if any brands deprecates a lot with conditions.
3. The most popular categories and subcategories should be scrutinized for any kind of information.
4. A feature of N-grams shall be taken in account as in the Personalized Cancer Diagnosis course case study but instead of direct Vectorization may be a feature for some ngrams can be added to keep resources in check.
5. There is a lot of scope for imputation of values in a textual feature as this challenge is NLP intensive. All the possible approaches should be given a fair try.
6. The most enriched feature is presumably the item description. Every kind of text encoding and their impacts shall be taken in account like BOW, TF-IDF, W2V.
7. A debugging approach to the model using failure cases should be kept in mind.
8. As this is a resource metered competition, constant garbage collections and pipelining must be a part of practise.
9. Initially predictions can be done with xgboost and/or lstm models, but resources will determine if a shift to LGBM, CNN, simple MLP or new light ensembles model is needed.

**Notes when you build your final notebook**:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar
2. You should not read train data files
3. The function1 takes only one argument “X” (a single data points i.e 1\*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
   1. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
   2. so in your final notebook, you need to pass only those two values
   3. def final(X):

preprocess data i.e data cleaning, filling missing values etc

compute features based on this X

use pre trained model

return predicted outputs

final([time, location])

* 1. in the instructions, we have mentioned two functions one with original values and one without it
  2. final([time, location]) # in this function you need to return the predictions, no need to compute the metric
  3. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y\_predict)

1. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data
2. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
3. Check this live session: <https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models>