Mercari Price Suggestion

Team - Inferno

Sujit Kumar
MT2020106
IIIT Bangalore
Sujit.Kumar@iiitb.org

Md Aqueeb Jawed
MT2020110
IIIT Bangalore
Md.Jawed@iiitb.org

Sounak Dey
MT2020081
IIIT Bangalore
sounak.dey@iiitb.org

***Abstract* - We are moving from retail to online shopping applications. Sellers in the online market find it difficult to sell their product due to the competitive market. Knowing the right price of their product will help sellers a lot to stay ahead in the competition.**

Our goal is to build a regression model that automatically suggests the right prices for products. We have made a model using traditional machine learning methods to predict the sale price of a listing based on information a user provides for this listing. Models try to predict the most accurate price of a given product based on the specific information provided about this product.

***Index-terms*- RMSLE, TF-IDF Vectorizer, one hot encoding, Ridge, Lasso, Light Gradient Boosting Machine(LightGBM)**

I. INTRODUCTION

DESCRIPTION - Over the past decade, the online sales market has grown exponentially. Online trading companies or internet shopping sites which sell a wide scope of things. In case of new products, the companies have their own strategies in deciding the price of the products but that's not the case with used products. Shopping websites need to suggest the prices automatically to their listed items. This recommendation of prices should be automated on the grounds that the quantity of things is tremendous to be finished by humans and furthermore it is extremely time-consuming and expensive. The challenge here is that the price of an item is subject to the kind of product and there are a huge number of various sorts of items. The price of similar items can change a ton depending on their image, condition of the item, specifications of the item, etc.

On the Mercari application, clients can pick whatever price they want when posting an item. This turns into an issue since certain items can't be sold in light of the fact that their posting costs are excessively high contrasted with the market cost and if the posting cost is lower than the market value, clients lose out.

Mercari, Japan's biggest community - powered shopping application, knows this issue profoundly. They'd prefer to offer estimating recommendations to sellers, yet this is intense in light of the fact that their sellers are allowed to put pretty much anything, or any bundle of things, on Mercari's marketplace.

As a solution, users can search Mercari for an item they plan to list, effectively performing market research. However, this is a lot of effort for the user, and this idea may not occur to new users. Therefore, listing becomes easier if we automatically display a suitable price for users when they list an item. So, almost all the used product websites must have their own price prediction models. Artificial Intelligence and Machine Learning has a big application here.

EVALUATION - It is a regression problem so we have a squared error related metrics as the error metric to minimize for making accurate predictions. As our product price has a very broad range from tens to thousands, we have to use a transformed version of squared error metric so that we do not get biased for accuracy on the products with high price.
Thus, we have used the "Root Mean Squared Logarithmic Error" (RMSLE)[1] as the error metric to minimize for this project. The mathematical formula for RMSLE is the following:

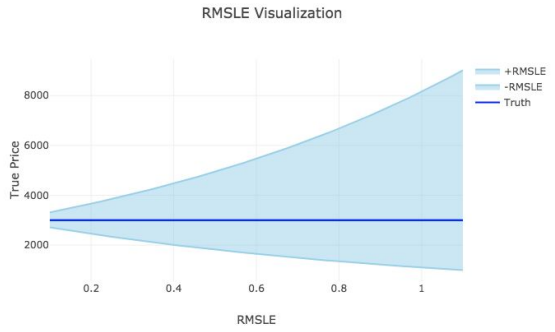
$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1) - \log(a_i+1))^2}$$

Where:

- ϵ: RMSLE score
- n: Number of predicted data cases
- p_i: Predicted price
- a_i: Real price
- $\log(x)$: Natural logarithm of x

The lower the score is, the higher the accuracy of the price suggestion function.

RMSLE Visualization
Below is a diagram showing the margin of error for RMSLE scores corresponding to a price of 3,000.



The horizontal axis represents RMSLE values. The vertical axis represents the error range of RMSLE values.
This is a logarithmic metric, so for a RMSLE value of 1.0:
– Real price: 3,000
– Range of estimated prices: 1,103 ~ 8,156

II. DATASET

The dataset used in the project is a part of the Mercari Price Suggestion Challenge dataset which was hosted on Kaggle.

It consists of 2 files:

- ***train.csv***: training data set
- ***test.csv:*** testing data set

Each data point in the training data consists of 8 columns, namely:

- ***train_id -*** the id of the listing

- ***name*** **-** the title of the listing.
- ***item_condition_id*** **-** the condition of the items provided by the seller
- ***category_name*** **-** category of the listing
- ***brand_name*** – brand of the item
- ***price*** **-** the price that the item was sold for.
- ***shipping*** **-** 1 if the shipping fee is paid by seller and 0 by buyer
- ***item_description*** **-** the full description of the item.

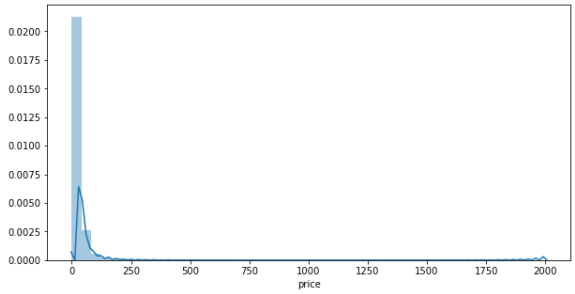
In the test data set only the price column is missing which is the target variable that we will predict. There are 1037774 data points in the training data and 444761 data points in the testing data.

III. DATA VISUALISATION

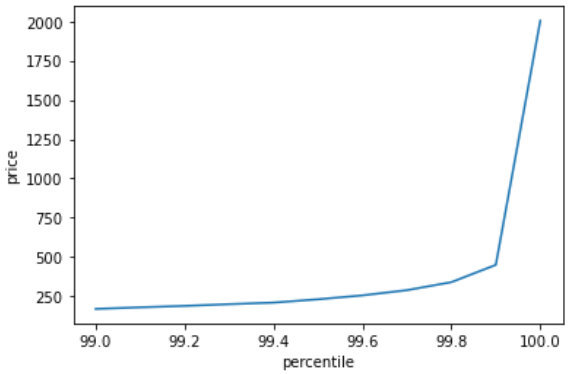
Getting insight about data is a very crucial task before moving to any predictive modelling.

Price

Price distribution is extremely skew and it ranges between 0 to 2004 with mean 26.7 and standard deviation 26.73.

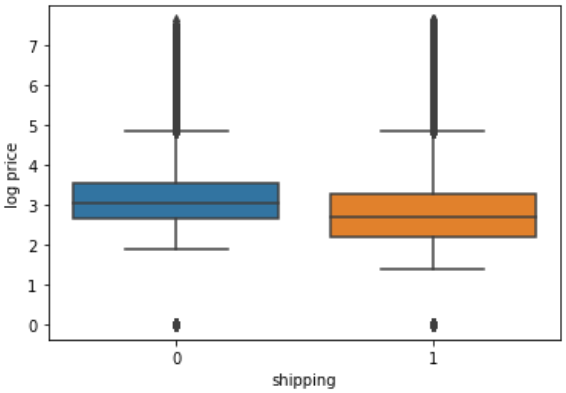


It can be noted that 99.9 percentile value is around 450 and inflection point occurs after that.



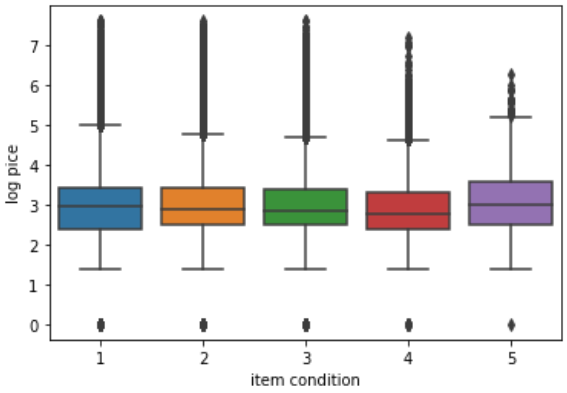
Shipping

Shipping status is binary i.e 0 and 1 where 0 represents buyer have to pay shipping charges and 1 means shipping is free. Around 55% of products come under 0 i.e buyers need to pay for shipping from their pocket. And also a product with no shipping charges are marked price higher

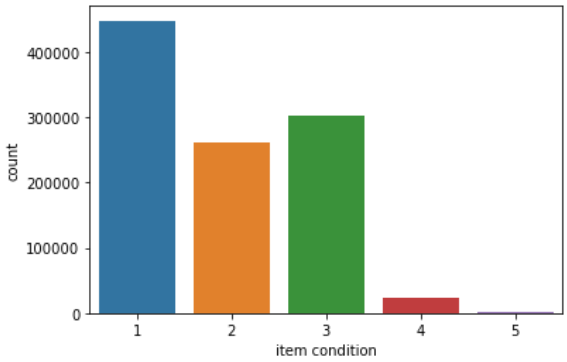


Item condition

From the box plot, we can see that the difference in prices are statistically significant too. No missing values have been found in the item_condition.

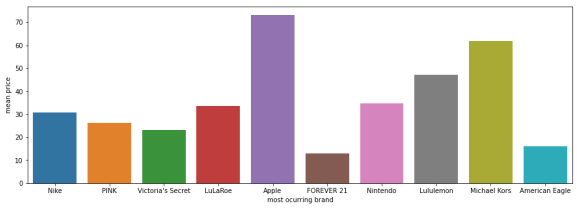


Products with item condition 1,2,3 have higher frequency compared to 4 and 5. There are very few products with item condition five.



Brand

Brand feature contains around 42% null value this may be due to human error or just they are brand less. Proenza Schouler is the most expensive brand across all brands with a mean and median price of 645. Apple products are the most occurring brand with the highest mean value.



From word cloud, we can see brands like Michael Kors, Victoria Secret, American Eagle etc are the most occurring brand.

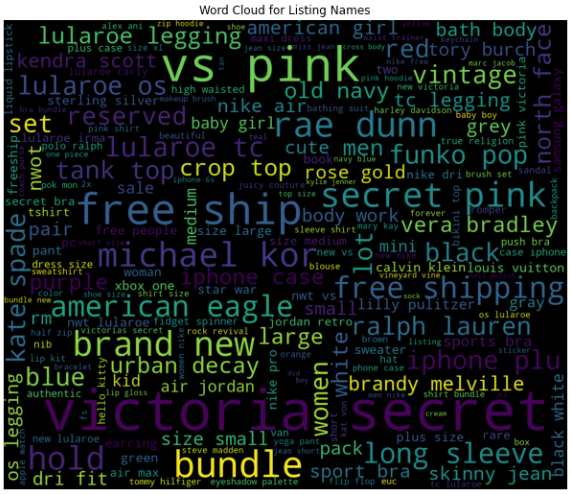
Category

We have 5 levels of category in which level 4 and 5 are mostly null.

By looking at word cloud we can see most of the product are of women, beauty, makeup etc

Names

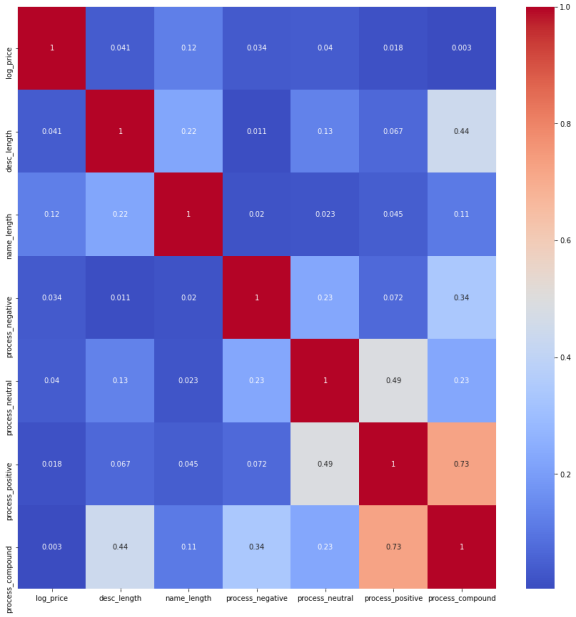
There is an interesting finding from the word cloud of names. Some brand names appear in the name feature. So we can use it to fill missing values in brand_name. Also



Item Description

Words like free shipping, brand new, new etc seem to be used to lure the customer.

Here is a heatmap between log price, desc length, name length and sentiment scores. We find there is no correlation between price and these features.



IV. FEATURE EXTRACTION AND PREPROCESSING

It is very common to find missing values arise due to many factors which are not in direct control. Sometimes due to the ways, the data was captured. In some cases, the values are not available at all for observation.

Item description, category name and brand name contain some missing value. We will fill the item description as “No description yet” and category as “other/others/others”. For the brand we will use the name to fill it and the value which remains null after this fill as ‘locals’.

We will split the category by ‘/’ and consider only the first three levels for training and ignore the rest due to null values.

We have to do basic text processing on name and item description i,e-

1. Converting all words to lowercase.
2. Removal of stop words
3. Removing punctuation and special characters.
4. Removing unwanted multiple spaces
5. Handling Alpha-numeric values and so on.
6. Lemmatizing

We will merge feature processed item description, name and three levels of the category to make a new feature ‘description‘ and we will produce ‘new name’ by merging name and brand name and drop original.

Ml model is not designed to handle string data so we need to encode them.

About TF-IDF Vectorizer: It is a numerical statistic that is intended to reflect the importance of a word in a document of a collection or corpus. TF-IDF vectors are calculated in the following way:

1. Calculate term frequency (TF) in each document.
2. Calculate the inverse document frequency (IDF): Take the total number of documents divided by the number of documents containing the word.
3. Iterate each document and count how often each word appears.
4. Calculate TF-IDF: multiply TF and IDF together[2]

We will use tf idf to encode our description feature with max features 50000,min_df =30 , and n gram (1,2).

For ‘newname’ we will use a count vectorizer. Idea behind count vectorizer[3] is very simple. Create a vector that has as many dimensions as your corpora has unique words. Each unique word has a unique dimension and will be represented by a 1 in that dimension with 0s everywhere else.

We will simply one hot encode[4] item_condition and shipping. Now we will hstack[5] all our features.

Now we are ready for model training.

V. MODELS

We have tried three models :

a) Lasso Regression
b) Ridge Regressor
c) Light GBM Classifier

1. **<u>Lasso Regression</u>**:
   Least Absolute Shrinkage and Selection Operator (LASSO), is an LR technique which performs regularisation on variables in consideration.

   The Lasso minimises the residual sum of squares to the sum of the absolute value of the coefficients

being less than a constant. Because of the nature of this constraint, it tends to produce some coefficients that are exactly 0 and hence gives interpretable models.[6]

It uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean. The lasso procedure encourages simple, sparse models. This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

The mathematical equation of Lasso Regression:-

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

= *(Residual Sum of Squares)*
+ *λ*(Sum of the absolute value of the magnitude of coefficients)*

Where,

- ***λ*** *denotes the amount of shrinkage.*
- ***λ = 0*** *implies all features are considered and it is equivalent to the linear regression where only the residual sum of squares is considered to build a predictive model*
- ***λ*** *= ∞ implies no feature is considered i.e, as λ closes to infinity it eliminates more and more features*
- *The bias increases with increase in λ*
- *variance increases with decrease in λ*

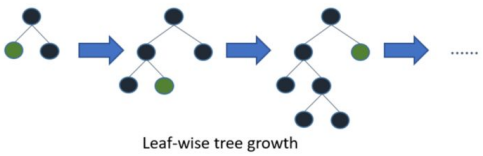
We have trained lasso with alpha = 0.0001 and fit_intercept=false. We got RMSLE score 6533896619381587

2. **<u>Ridge Regression</u>**:
Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors. It is hoped that the net effect will be to give estimates that are more reliable.

Ridge regression belongs to a class of regression tools that use L2 regularization. The other type of regularization, L1 regularization, limits the size of the coefficients by adding an L1 penalty equal to the absolute value of the magnitude of coefficients. This sometimes results in the elimination of some coefficients altogether, which can yield sparse models. L2 regularization adds an L2 penalty, which equals the square of the

magnitude of coefficients. All coefficients are shrunk by the same factor (so none are eliminated). Unlike L1 regularization, L2 will not result in sparse models. We have trained our ridge regressor with parameter alpha=1, solver as sag and fit_intercept false and we got RMSLE 0.4935887821371837;

3. **Light GBM**:
   It is a gradient boosting framework that utilizes a tree based learning algorithm. LightGBM grows trees vertically implying that the tree grows leaf-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm.
   LightGBM can deal with the huge size of data and takes lower memory to run.[7]



Leaf-wise tree growth

Parameter	Description	Value
n_estimators	Number of boosting iteration	5000
num_leaves	maximum number of leaves each weak learner has	100
max_depth	This parameter control max depth of each trained tree	1400
application	Used to specify the type of ML model	regression
metric	metric(s) to be evaluated on the evaluation set(s)	RMSLE
verbosity	controls the level of LightGBM's verbosity	-1
n_jobs	number of threads for LightGBM	-1

RMSLE score for LightGBM was 0.436 .

VI. CONCLUSION

We conclude that we were able to come up with an efficient model with a RMSLE score of 0.436, to predict the price of a given product online, based on the various information and descriptions we have at our disposal about the product.

VII. ACKNOWLEDGMENT

We would like to thank Professor G. Srinivasaraghavan for the highly detailed lectures on various topics which helped us understand what we were actually doing, and a special thanks to our Machine Learning Teaching Assistant Amitesh

Anand for giving us the opportunity to work on the project and help us by giving us ideas and resources to learn from and for helping us out in the initial stages and whenever required on numerous occasions.

VIII. REFERENCES

[1] Sharoon Saxena "What's the Difference Between RMSE and RMSLE?" [Online: posted 26-june-2019] Available:\url[https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a]

[2] Kaitlyn, "Calculating tf-idf with python," 2017, [Online; posted 3-June-2017]. [Online]. Available: \url{https://methodi.ca/recipes/calculating-Tf-idf-python}

[3] Hunter Heidenreich, "Natural Language Processing: Count Vectorization with scikit-learn" [Online: posted 24-Aug-2019] Available:\url[https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e]

[4] Jason Brownlee, "Why One-Hot Encode Data in Machine Learning?" [Online: posted 30-Jun-2020] Available:\url[https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/]

[5] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed <today>]. [Online]. Available: http://www.scipy.org/

[6]Dinesh Kumar, "A Complete understanding of LASSO Regression" [Online: posted 4-Sept-2020] Available:\url[https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/]

[7] Pushkar Mandot, "What is LightGBM, How to implement it? How to fine tune the Parameter?" 2017 [Online; posted 18-Aug-2017] Available: \url[https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc]