Applied Data Science - Spring 2020

Price Prediction using Product Description

Team C

Sparsh Tekriwal
Eashan Chawla
Pranav Banthia

Applied Data Science: Spring 2020

Price Prediction Using Product Description

Executive Summary

Online marketplaces such as Mercari allow sellers to freely set their ask price, but it can be hard for sellers to determine how much their possession is really worth. With an average monthly transaction of $100 million, the platform has more than 5 million monthly users. Small details about the product can mean big differences in pricing based on their brand, condition of the item, specifications of the item and so on. For example, a St. John's Bay Pullover is typically priced in the range of $10 - $20 whereas a Vince branded sweater could easily sell for over $300! And it is quite hard to guess the identity based on only the product description.

Not knowing the optimal listing price, a seller could end up under-valuing, or over-valuing the product. The former could lead to the user losing out on potential profits. In the latter case, the seller lists too high a price and eventually has to reduce it (since pragmatically users show less inclination towards overpriced products). Suggesting an appropriate mark price will help improve and speed up the selling process, which would not only benefit the sellers, but also the business by increasing the transaction volume.

While the correlation between an abstract user-generated description of a product and its price may sound absurd at first, we have tried to frame this text-based prediction as a machine learning problem. As machine learning models cannot understand raw text data, we employ various natural language processing techniques to generate numerical representations of these inputs. After experimenting with a variety of model types, we found the right balance of interpretability and performance in a regularized linear regression approach with Ridge penalty.

With a R-squared value of 0.45, our final model is able to successfully explain 45 percent of the response variable variation in the data. Additionally, around 50 percent of predictions made by our model are within 30 percent of the true transactional price of the product. Only scenarios of over and underestimation include ambiguous product descriptions given by sellers. Our solution gives extremely accurate results for products with unequivocal item description.

1. Data description and factors affecting price prediction

The dataset for this project consists of 1.4 million records of product listings, in english language on the e-commerce website, Mercari, that presently operates in Japan and the United States. The values are in a tab-separated format with each row having the following attributes/features for any given product:

- *train_id* - unique identifier of the product
- *name* - the title of the product listing
- *item_condition_id* - the condition of the product provided by the seller
- *category_name* - Three hierarchical category levels as maintained on the platform
- *Brand_name* - brand name of the product
- *price* - the price in USD that the product was sold for.
- *shipping* - 1 if shipping fee is paid by seller and 0 if paid by buyer
- *item_description* - the full description of the item.

Input features: train_id, name, item_condition_id, category_name, brand_name, shipping, item_description. Branded, newer products and higher price of similar products directly affect the price prediction.

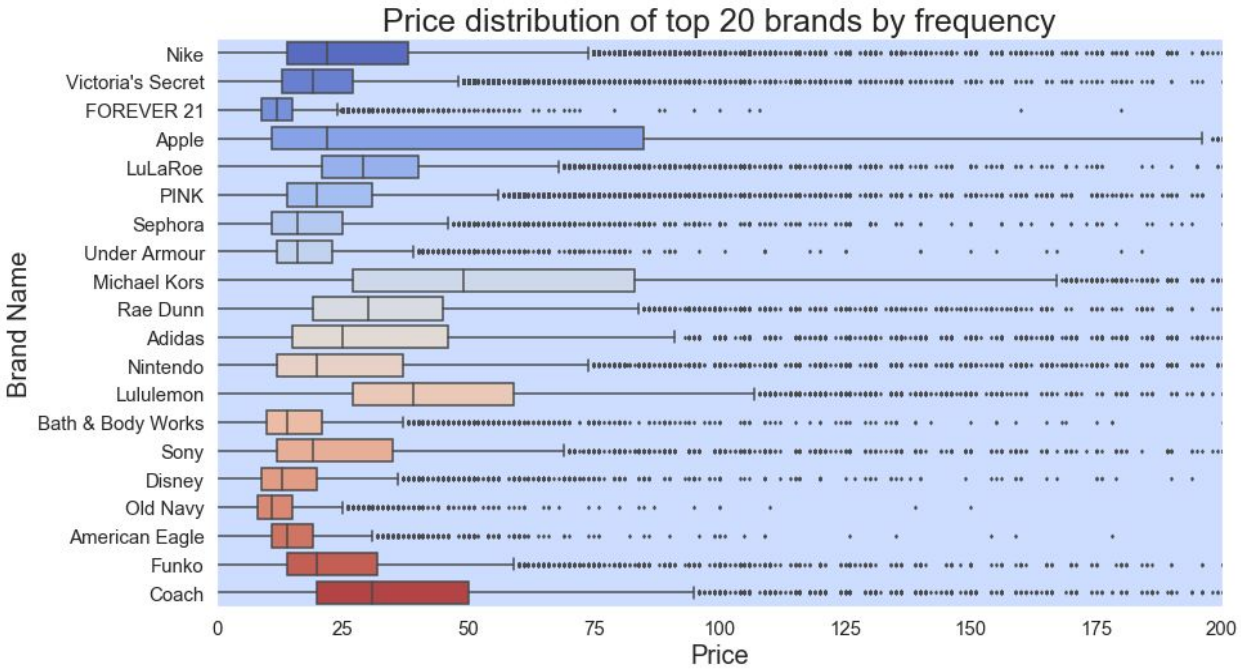
Target Variable: price

Below is a sample product listing. Factors affecting the price (in green) are as marked in blue :

Name	**Item Description**	**Brand**	**Category Name**	**Condition ID**	Shipping	**Price**
Girls Nike Pro shorts	Girls Size small plus green. Three shorts total.	Nike	Sports & Outdoors / Apparel / Girls	3	0	19

All potential sellers on the website have to input the product name, condition of that product, category, brand, shipping and the description of the item they wish to sell. Here, that the price is something our system will predict for them based on the other fields they have provided as input.

Shown below is an illustration that helps us understand the price distribution of the most frequently sold brands on the website. Amongst these are some of the premium brands like Michael Kors, Apple, Lululemon, and Coach - whose items are generally much more expensive (trend seen from the upper quartile of their respective distributions).



2. Experimental Setup

2.1 Feature Engineering and Preprocessing of Text

First, we preprocessed our text input features - “name” and “Item_description”. This is a standard practice when dealing with text data and is needed to be performed before any model is applied.

2.1.1 Preprocessing

Text Cleaning:

This step involved removal of special characters, emoticons, punctuations, escape sequences, and stop words. Numbers and alpha-numeric were retained because they contain useful information. Eg: iPhone 7, XBox 360, MacBook Pro 15" 512 SSD i7 3.5GHz 16GB etc.

Converting all text to lowercase:

Users often tend to give inputs in various cases like Iphone, iPhone, iphone, IPHONE, etc. Such variation in case structure might introduce redundancies in our model and ultimately affect the

accuracy of our predictions. Hence, in an effort to push our system towards achieving accurate predictions we convert all the text input to lowercase.

Lemmatization:

It is very natural for people to use different 'inflected' forms of the same word when providing descriptions of their items. Hence, to account for these we performed Lemmatization which is the process of grouping together the inflected forms of a word so they can be analysed as a single item. This helped us deal with some of the redundancy in our feature space.
Examples:

- reducing, reduces, reduced, reduction --> reduce
- n't --> not
- 've --> have

2.1.2 One Hot Encoding Categorical Columns

To provide meaningful input to our machine learning model, we encoded the categorical columns into a One-Hot Encoded sparse vector as shown below:

train_id	general_cat
0	Men
1	Electronics
2	Women

→

train_id	Gen_Cat_Electronics	Gen_Cat_Men	Gen_Cat_Women
0	0	1	0
1	1	0	0
2	0	0	1

2.1.3 Bag of Words - 1, 2-grams (with and without Tfidf)

Similar to categorical features, we need to vectorize our preprocessed text columns. This is done with the help of methods such as Term frequency-inverse document frequency (tf-idf)), and Count Vectorization, and Word Embeddings.

Tf-idf which discounts the emphasis of common words did not lead to an improvement in results for our model because the length of our documents (description) was quite short.

We also transformed our text into a 300 dimensional word2Vec vector using the Google news corpus. However, this was not particularly helpful for our use case.

For products with descriptions containing text such as “brand new”, “never used”, “Michael Kors”, “Salvatore Ferragamo”, etc we make use of bi-grams to generate meaningful feature names.

2.1.4 Numerical Features

Having numerical features, in addition to one-hot encoded features could skew the results of the model, giving more importance to the higher valued numerical features. Therefore, we normalized our numerical feature item_condition_id using MinMaxScaler method.

2.2 Machine Learning Models

2.2.1 Linear Regression:

Given an input vector $x \in Rm$,(independent variables or predictors), we find a prediction $\hat{y} \in R$ for the recommended selling price of a product $y \in R$ using a linear regression model: $\hat{y} = \beta 0 + x1\beta$ where $\beta 0$ and β are the parameters to estimate. Usually, the parameters are learned by minimizing the sum of squared errors.

2.2.2 L2 Regularized Regression (Ridge):

An extension of Linear Regression, here the loss function is modified to minimize the complexity of the model measured as the sum squared value of the coefficient values. This gives the effect of shrinking the coefficient, and the complexity, reducing the response of coefficients with minor contributions to almost 0. Ridge also allows us to strive for high explanatory value.

Hence, it is more interpretable for a Business Analyst trying to understand the feature associated with a particular ridge coefficient and its importance. (Refer section 3 for an illustration of ridge coefficient as compared against the features).

2.2.3 LGBM and XG Boost:

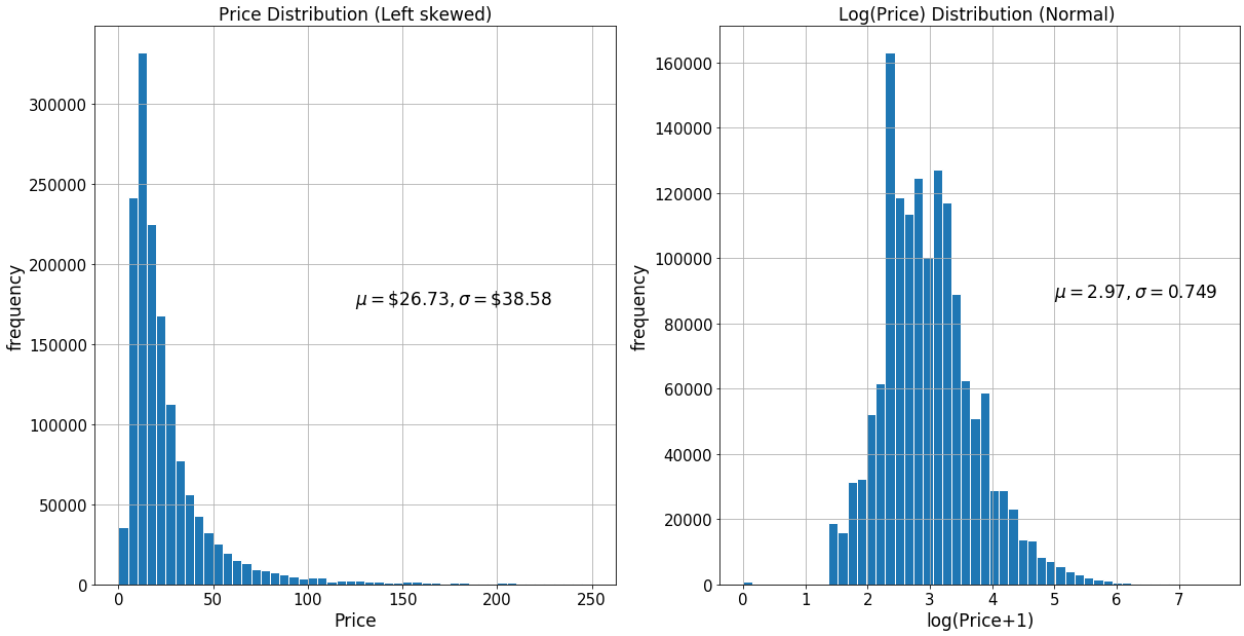
LightGBM and XG Boost are tree based ensemble methods which tend to perform very well for unstructured data. In our case the user inputted item description is an example of unstructured data. Hence, we used this tree based gradient boosting approach to assess the accuracy of

our system. But the results were not supporting our hypothesis and we did not achieve better accuracy using these methods.

2.3 Performance Metrics

As the distribution of raw prices in our data set is skewed towards zero, we used a log distribution which closely resembles a bell shape. The implementation is done as follows :

```
y_log = log (y + 1)
model.fit (X, y_log)
prediction_log = model.predict(X_test)
prediction  = exp(prediction_log) - 1
```



2.3.1. Root Mean Squared Logarithmic Error (RMSLE)

$$\text{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$$

The root mean squared logarithmic error is calculated using the above formula. Here $\hat{y}_i$ is the predicted price of the product, a_i is the true price of the product, and n indicates the number of

products in the test set. After evaluating multiple metrics, we found RMSLE to be best suited for our purpose. The following are some observations that influenced this decision:

RMSLE considers only relative error and not the absolute error (because of the logarithm term) —Mercari has over 10 million+ products. Every product has a different price. For example a pair of socks are sold on an average for $5-$10 whereas a new iphone might cost around $800-$1000. Hence, a price prediction of $9 against original price of $10 and a prediction of $900 against $1000 are both having the same relative errors. This helps business understand overall error in a more interpretable way.

RMSLE penalizes more when you are under-estimating rather than over-estimating

2.3.3. R squared

The R squared metric provides an indication of the goodness of fit of a set of predictions to the actual values. It indicates how well a particular model performs as compared to the default model. This is a value between -inf to 1 for no-fit and perfect fit respectively. We achieved R-squared of 0.45 which explains that our model captures 45 percent variance in the prices as compared to the default model.

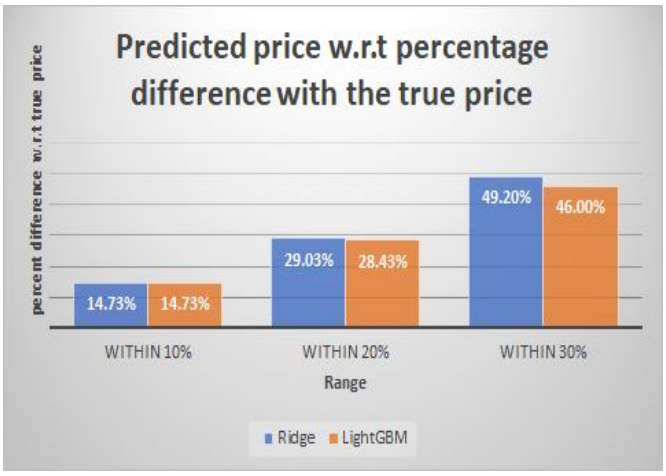
2.4 Tuning Model Hyper-parameters using GridSearchCV

We tuned the hyperparameters for our best performing model - Ridge
Parameters = {"alpha": [0.0001, 0.001, 0.1, 1, 10, 20, 30, 50, 100]}
Best Parameter = {“alpha” = 20}. Here alpha is a measure of how much we would like to penalize large values of coefficients.

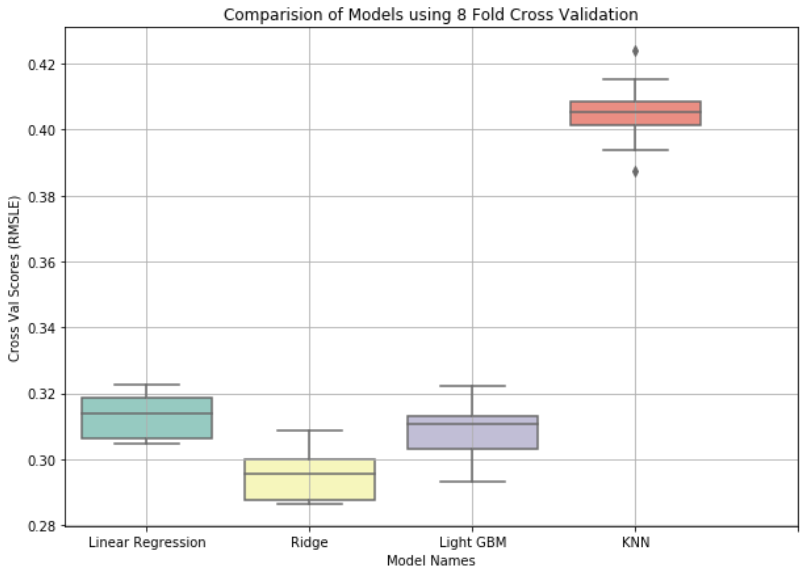
3. Interpretation of Results

When we saw items with big errors, and tried to analyze why our models were wrong in those particular cases. And we noticed that one reason why our model was wrong was because people were selling bundled items for example 5 lipsticks , 10 face creams, 5 brushes etc.

name	Ridge	Light GBM	Actual Price
Gymboree slides size 9	19.70	21.14	16.0
Decor hanging	13.90	15.84	12.0
DAISO CLAY BLUE	5.85	13.31	9.0
Cute Cloth diapers O/S	16.82	13.84	14.0
Winter Boots And Jacket	26.30	24.68	30.0
Nintendo ds lite	29.43	22.82	21.0
e eyeshadow pallet #25A	40.40	41.89	44.0
kids adidas	23.02	23.95	36.0
Boutique top	9.47	14.18	10.0



Sample Price predictions of our system



3.1 Baseline Model (Mean prediction):

The baseline model predicted an average price of ***$26.84*** for all products. R-squared for the baseline model is 0, and we expect that any model that learns something new/in a better way, will have an R-squared greater than this. The RMSLE of this model was 0.745.

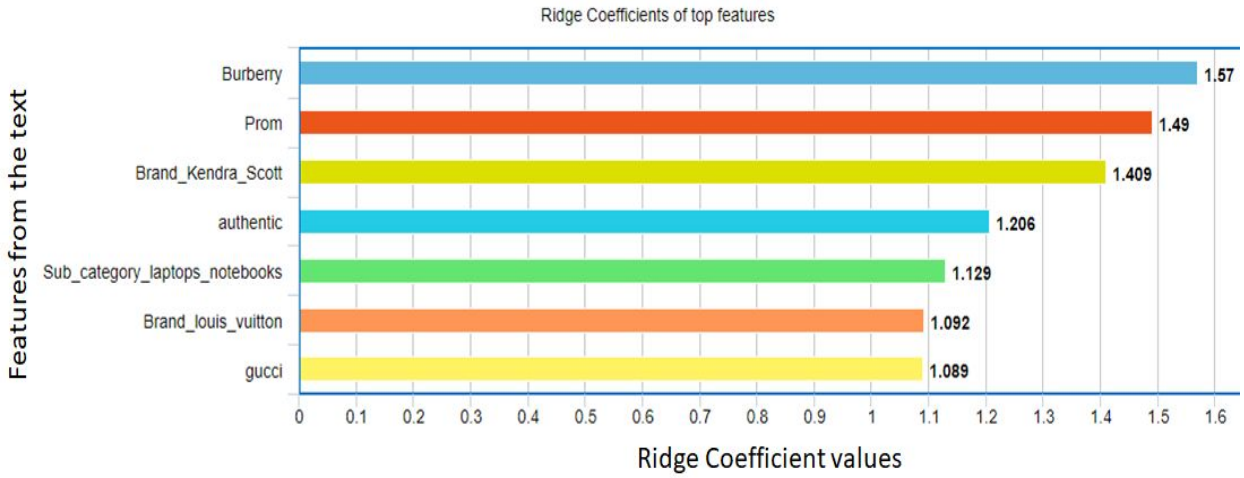
3.2 Linear Regression:

Performance for our linear model indicated an R-squared of 16.9%, indicating that even though it's performance wasn't that great, it was still learning something.

3.2 Ridge Regression:

Ridge regression performed the best out of all the models we tried. It gave us an R-squared of 0.371 or 37.1%, and a cross validated RMSLE of 0.39.

Upon inspection of top coefficients for ridge regression, we noted that ridge seemed to give importance to the right variables as shown below: We see that Ridge gives the most importance to features (brands) like Burberry, Louis Vuitton, Kendra Scott, among others. This makes intuitive sense as something that is labeled as “Burberry” or “Louis Vuitton”, will be more expensive than any other non branded product.



3.2 XG Boost and LightGBM:

Performances for XGBoost was as follows: R-squared of 34.8% and CV RMSLE score of 0.33. Performance for LightGBM was as follows: R-squared of 36.7% and CV RMSLE score of 0.39. Even though they have good scores, they seemed to be giving importance to wrong coefficients. In addition to this, tree based models give us feature importances rather than coefficients, making it difficult to interpret them and compare with linear models.

4. Risk and Mitigation Strategies

Susceptibility to outliers: Certain items in the dataset which would have a very high price value can actually skew the model if you evaluate on RMSE because of the squared error penalty in RMSE , whereas the RMSLE would only incur a slightly higher penalty for high value cars

compared to low value cars because RMSLE has a logarithm term in the metric that cushions this effect.

Overestimation / Underestimation: *A common problem faced in the field of applied data science is the susceptibility of models to over or under-estimate/fit the data that they are trained on. A solution to this implemented by us was cross validation.*

Language (Reproducibility problem in new geographies. Cold start): Mercari is an ecommerce platform operating in Japan and the United States. Mainly operating in Japan, we would have to ensure that our solution is able to comprehend and work with that language, which currently it cannot. This can lead to a cold start problem.

Numbers in Text: Since the description of the product is a free text description given by the user, we found that in a lot of cases, it contained random numbers in text.

List vs transaction price: Prices in the dataset used by us are the prices at which a product was listed by a particular user, and not the price at which it was sold. Due to this, there is a lot of variation in prices entered by users for similar items, that could potentially skew the results. A simple solution to this would be for Mercari to replace the list prices with actual transaction prices before deploying the model to an endpoint API.

5. Recommendations for Future Work

Deployment on PySpark: Vectorization of text can lead to an extremely large number of features. As our training dataset grows with time, training models of such a high dimensionality would require a lot of computational power. PySpark can significantly accelerate analysis by making it easy to combine local and distributed data transformation operations while keeping control of computing costs. In addition, the language helps to avoid always having to downsample large sets of data.

Exploring Deep Learning methods: It is not obvious how to effectively represent unstructured text data with features for this particular learning task. Neural Networks are able to create multidimensional mappings of complicated, nonlinear relationships in features, which is not possible with classical machine learning models. Since learning time is not a concern for the platform (can be done over extended periods), but predictions must be quick, neural networks could give significantly better results and are worth exploring.

References:

Brownlee, J. (2019, August 7). A Gentle Introduction to the Bag-of-Words Model. Retrieved from *https://machinelearningmastery.com/gentle-introduction-bag-words-model/*

Brownlee, J. (2014, January 1). How to Use Machine Learning Results. Retrieved from *https://machinelearningmastery.com/how-to-use-machine-learning-results/*

Shahzad, Q., & Ramsha, A. (n.d.). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications, 181(1). Retrieved from *https://www.researchgate.net/publication/326425709_Text_Mining_Use_of_TF-IDF_to_Examine_the_Relevance_of_Words_to_Documents*

Hickman, B., Bodoh-Creed, A., & Boehnke, J. (n.d.). Using machine Learning to predict price dispersion. Retrieved from *https://scholar.harvard.edu/files/boehnke/files/bcbh_machine_learning_price_dispersion.pdf*

Lemmatization. (n.d.). Retrieved from *https://en.wikipedia.org/wiki/Lemmatisation*

Saxena, S. (2019, June 26). What's the Difference Between RMSE and RMSLE? Retrieved from *https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a*

Brownlee, J. (2016, March 21). Overfitting and Underfitting with ML Algorithms Retrieved from *https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/*