**Mercari Price Prediction with Natural Language Processing**

**--INFO7390 Final Project**

Jiali Cheng 001822320, Sicheng Zhang 0018

1. Introduction

In this project, we are dealing with Mercali price prediction, which is a Kaggle competition. Mercali is a C2C online trading platform where customers can post their products with some description and wait for the buyers. While setting a right price for a higher and quicker sale comes to the problem. In this process, Mercali, the platform, will help the seller set a proper price based on the features and descriptions provided by the seller. And we will use the Mercali sales data to build a machine learning model helping the platform achieve this goal. Then we will deploy this model on the cloud so that everyone can access this prediction model if they need with a REST API.

In the dataset, there are 6 common features and 2 columns of plain text. 6 features appear to have been a little less for an entrepreneurial prediction model with a huge amount of user and we will see later the score of this raw model is relatively low. So we leverage some NLP techniques to extract features from the text, including Bag-of-Words, Word Embedding and TD-IDF, and build a more powerful model.

For the prediction model, we use statistical method, ridge regression, random forest and neural network for prediction. And we compare they results both between various models and between whether using the text features or not.

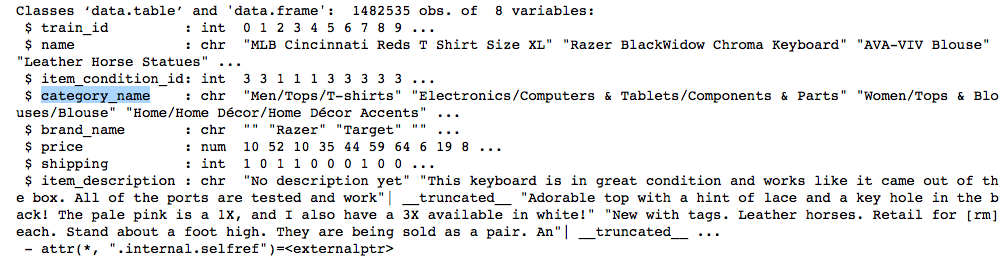
1. Data Engineering
   1. Data Ingestion

We get this data from Kaggle in CSV format. We import it with Jupyter Notebootk.

* 1. Exploratory Data Analysis

(1) General

One thing that we supposed to do before we go deep into EDA is having a general look of the data set.



By look through all the eight columns, there are two of them draw my attention tightly. First is the ‘item\_description’ which need NLP to handle with. Another is the ‘category\_name’ which mostly consists of three levels, seems need some tricky strategies.

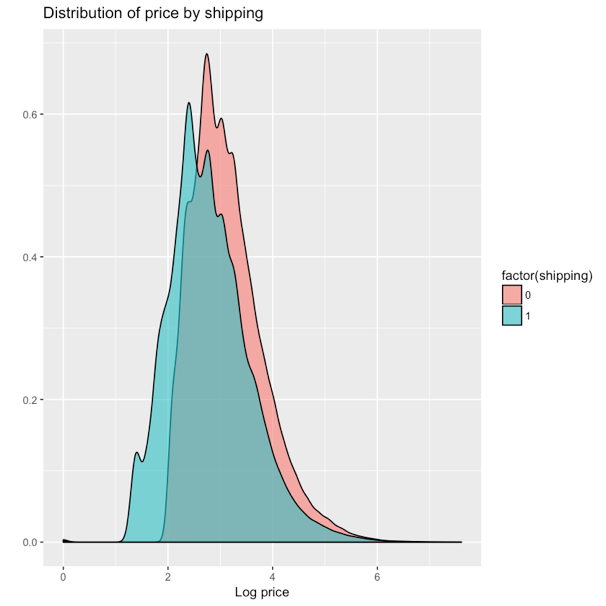
(2) Price

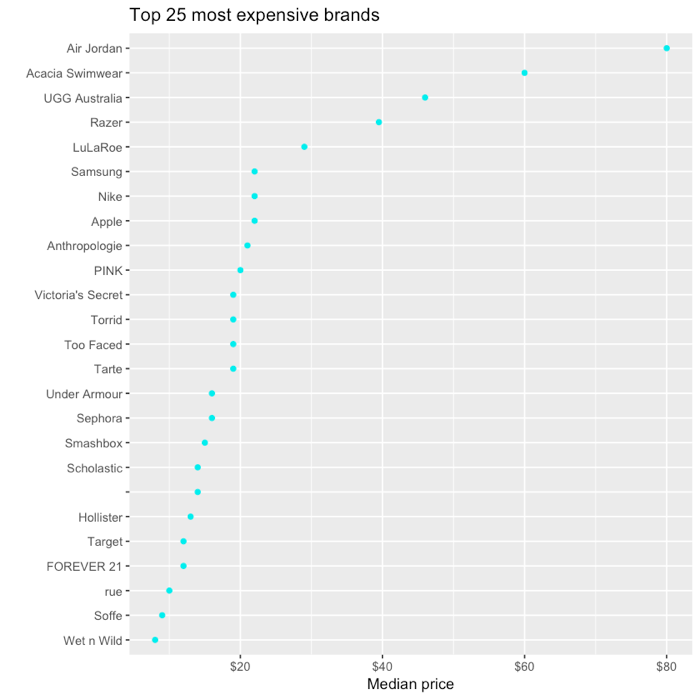
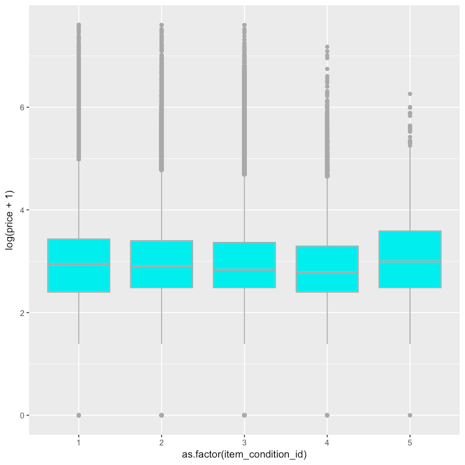
Primarily, let’s have a look of our target—price. The price ranges from 0 to 2009. Because price is likely skewed and because there are some 0s, we’ll plot the log(price+1).

As the image shows that over 80% is between 2 to 4.

(3) Item Condition

The item condition ranges from 1 to 5. There are more items of condition 1 than any other. Items of condition 4 and 5 are relatively rare. It’s not clear from the data description what the ordinarily of this variable is.

Our assumption is that conditions 4 and 5 are more likely to be good conditions because they are so rare. We can try and verify this. If a higher item condition is better, it should have a positive correlation with price. Let’s see if that is the case.

Looking at the average price by condition shows a relationship that is not quite as neat as I had hoped. Condition 5 clearly has the highest price, however condition 1 has the next-highest price, followed by condition 2, then 3, then 4. Consider with the numbers of items for each condition, we can easily know that the condition is decreasing through 1 to 5. The reason that the price of condition 5 is higher than other’s is the uncertainty cause by the data insufficient.

(4) Shipping info

My initial thought is that items where the shipping fee is paid by the seller will be higher-priced. However, there are numbers of conflating factors. This may be true within specific product categories and item conditions, but not when comparing items on the aggregate. Let’s see.

From the plot, we can ensure that the price will be higher if shipment fee is paid by seller.

(5) Brand

The Air Jordan and Acacia Swimear brands are by far the most expensive brands, with a median price of $80 and $60 respectively.

Nearly half of the items don't have brands. The proportions of items that have brands vary in different categories. For example, nearly all handmade items don't have brand names, of course.

For brands, they are not in a hierarchical order and there are too many to be fitted in one graph. So, I plotted the count of top 10 most frequent brands for a rough look. Each brand contains items from 1 or more major categories. Not surprisingly, the top brands are dominated by women items except Apple and Nintendo.

(6) Category name

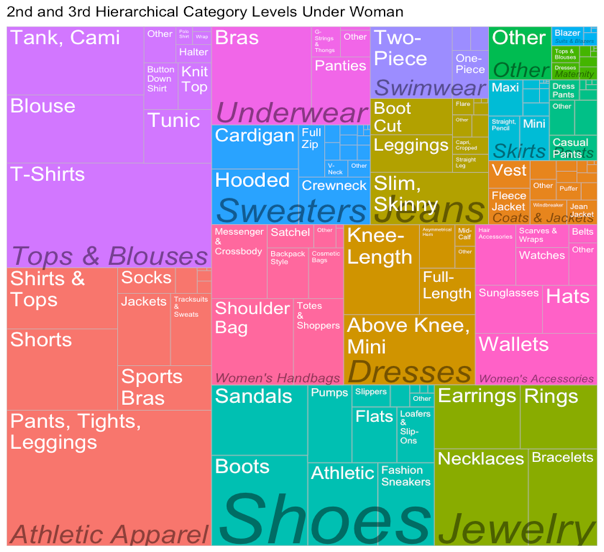
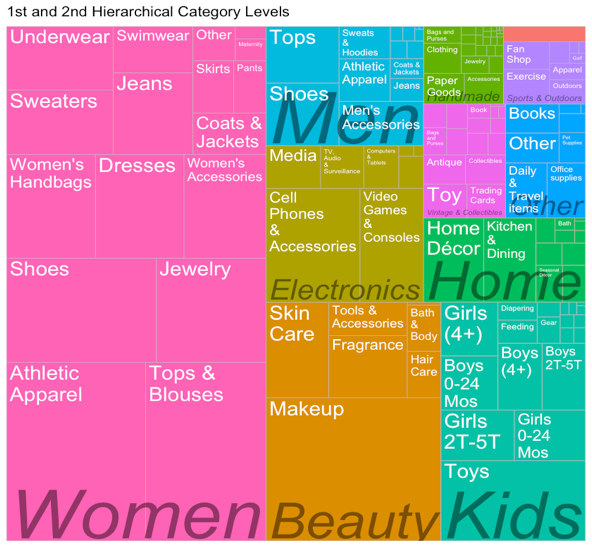
We notice that the category\_name is actually encoded as three or four hierarchical levels splitted by ‘/’. (Thanks to Abhinav Reddy Kaitha there are some items with four levels instead of three)

We can split the category names and store them into 4 columns. The major category (1st category) only has 11 levels and we can make distinguishable visualizations on them. From the 2nd level on the # of levels are too many to visualize.



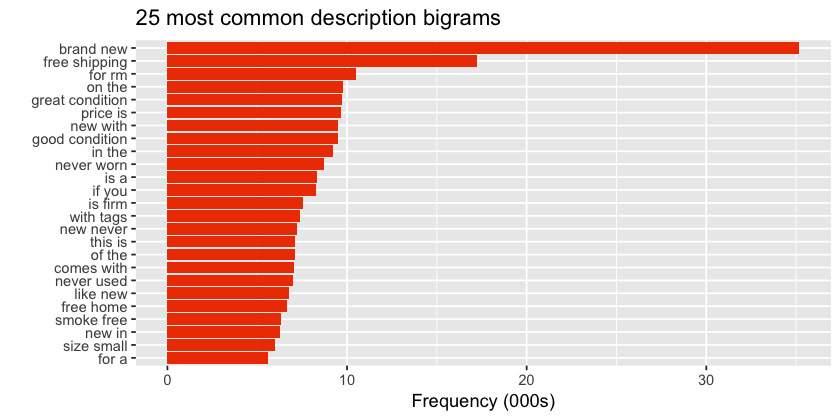
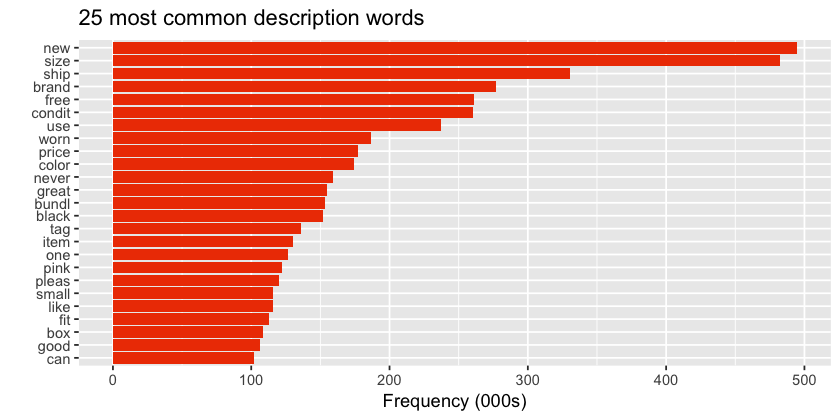
Most items only have three levels of categories. But the 4th level exists with 8 unique sub-categories and 4389 items. For modeling perspective, it may be fine to combine it with 3rd levels but for analysis purpose I extract and keep the 4th level here.

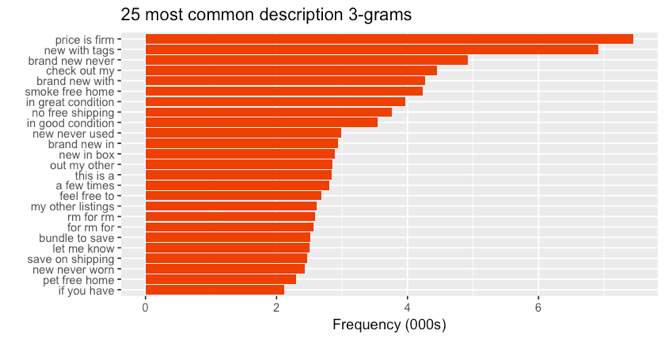
For visualization, we plot the 1st and the 2nd level of categories to show the distribution of the number of the retail items. We can see that the majority, nearly a half, items belong to Women category. So, we than plot the 2nd and 3rd layer of the Women category to have a deeper look.



(7) Item description

For item description, we have mainly three steps: remove the price description; remove English stop words, punctuation, and stem words; using N-grams to find out most common grams when N=1,2,3.





* 1. Data Cleaning and Preprocessing

We fill all the missing data with the string ‘missing’, which mean we take missing value as one of the categories. Then we encode the category data, which include the brand name and category of products.

1. Prediction without NLP
   1. Statistical Method

For this kind of data with a lot of category values, using group average first comes into our mind. We group the data based on the features of category, shipping information, decreases the data set we have to process.

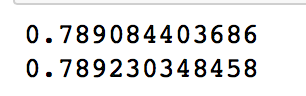
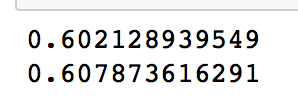
* 1. Ridge Regression and Random Forest

We use ridge regression instead of linear regression because: when there are too many correlated features in linear regression, their coefficients can be poorly determined and exhibit high variance. A wildly large positive coefficient can be canceled by a similarly large negative coefficient. By imposing a size constraint one the coefficients, this problem is alleviated.

We get the results below:

Random forest (left), meaning that the predicted price is in the range of 0.54-1.86 real price.

Ridge regression (right): meaning that the predicted price is in the range of 0.45-2.18 real price.



1. Prediction with NLP

Now we will leverage the information given by the columns of ‘name’ and ‘item description’. We tried different ways. One is extracting informative features from the text. Another way is to embed the words into vectors.

* 1. Feature Extraction

First we define the features to be extracted shown in the following figure. And we extract these features with regular expression.



Then we train the data with ridge regression and random forest and get ridge and get the results below:

Random forest 0.7739, meaning that the predicted price is in the range of 0.46-2.16 real price.

Ridge regression 0.5558: meaning that the predicted price is in the range of 0.57-1.74 real price.

* 1. Word Embedding

Word Embedding is a natural language processing model where word is represented as a vector. A sentence is the sum of all its words’ vector. We multiply the sum by a matrix and add some bias. Then we put the sentence vector into a neural network for training. There are different ways of embedding. We tried Bag of Words, Word to Vector and TF-IDF.

* + 1. Bag of Words

Bag of words counts the frequency of each word in the text and can be regarded as a one-hot embedding of words.

Since we are not focusing how to build a neural network, we use a neural network build by ‘noobhound’[[1]](#footnote-1)[1] from Kaggle.

When we try batch size of 20000 and epoch of 5, the result is 0.4528, meaning that the predicted price is in the range of 0.6-1.5 real price. When we try batch size of 10000 and epoch of 10, the result is 0.4393, meaning that the predicted price is in the range of 0.64-1.55 real price. We can see that decreasing batch size and increasing training epochs will increase the accuracy.

And the best log error on Kaggle is close to 0.4, meaning that the predicted price is in the range of 0.67-1.49 real price. Most people are getting results in 0.41-0.5.

* + 1. Word to Vector

Word to vector embed words in a more meaningful way. Words are transformed into vectors, or multi-dimensional spaces. And the direction of the vector the meaning of a word. For example, if we know ‘king’ points to ‘queen’, given the word ‘man’, we could get the vector of ‘woman’.

When we tried word to vector, the most difficult part is not the training part, but forming the sentence vector of a piece of description, which could take days and crash the whole system on the cloud. So we just use a little part of the data and train them. And we are getting a worse result of 0.7211 log error, which means the predicted price is in the range of 0.5-2.0 real price.

* + 1. TF-IDF

TF-IDF stands for Term Frequency – Inverse Document Frequency, which indicates the importance of a word to a document in a corpus. Similar with the weight factor when we select features in Neural Network and Random Forest, TF-IDF can help us filter the words that only appear few times and the words appear frequently but have no significant meaning.

TF-IDF also suffers from problem of an extremely large matrix, which would take days to train. So we just use a part of the data to train and get a result of 0.6237, meaning that the predicted price is in the range of 0.5-1.8 real price.

1. Deployment and Documentation

We deploy the model on Google Cloud Platform with Flask. And everyone can access our prediction model with a web service at the IP address we gave.

We deployed random forest as a web service, since neural network is too difficult to deploy and random forest takes category features better than regression.

To access the web service, navigate to the website and you will see this UI where there are four demoes that you can run. They are all real data from the dataset. Or you can input any data and description to let our model predict the price of your product based on the information you provide. We will just show one of the demoes below.

1. Conclusion

Natural Language Processing is giving better results versus non-NLP techniques.

The results are as below:

Non-NLP:

'Group Mean': 0.5785,

'Ridge(lsqr)': 0.8132,

'Ridge(sag)': 0.7891,

'random forest': 0.6021

NLP:

'Bag of Words(10 epochs)': 0.4393,

'Bag of Words(5 epochs)': 0.4528,

'Ridge(sag)': 0.7739,

'TF-IDF(5D)': 0.4393,

'Word Embedding(2D)': 0.4393,

'random forest': 0.5558

We can see that using natural language processing techniques we can extract many feature outside a simple piece of text. When doing word embedding, we embed the meaning of a word into a multi-dimensional vector. By doing this we can retain the information of a word. And by taking a non-linear combination, the model can learn some deep embedded information in the text.

Neural network gives the best results for text processing. This is because by taking some non-linear activation function in neural networks, the model will learn abstract information like grammar. Neural network is really powerful dealing with natural language processing.

Under same circumstances, random forest is always giving better results than regression. This is because most of the data are categorical features. Random forest deals with categories values better than regression.

ACKNOLEDGEMENT

We would like to thank Prof. Sri and TA Modani for their precious time and great help through this semester. We can feel that this course is unique to other academic course and let us engage in industrial jobs. We really learned a lot. Thank you!

1. [1] Special acknowledgement to ‘noobhound’, a contributor of Kaggle. We use his neural network model for training the embedded vectors. [↑](#footnote-ref-1)