**Project Report – Mercary Price Suggestion Challenge**

1. **Business Understanding**

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.

[Mercari](https://www.mercari.com/), Japan’s biggest community-powered shopping app, knows this problem deeply. They’d like to offer pricing suggestions to sellers, but this is tough because their sellers are enabled to put just about anything, or any bundle of things, on Mercari's marketplace.

**Problem Statement**

Build an algorithm that automatically suggests the right product prices.

1. **Data Understanding**

Mercary Price Suggestion Challenge contains following two datasets

Train.tsv

Test.tsv

1. **Load Packages –**

Load library ggplot for visualizations and data.table for reading the train and the test data.

library**(**"ggplot2"**)**

library**(**"data.table"**)**

1. **Reading Data –**

Using fread() method for reading train and test data into R, for faster read and lesser RAM usage.

train **<-** fread**(**input **=** "../input/train.tsv", header**=TRUE**, stringsAsFactors **=** **FALSE**, sep**=**'\t'**)**

test **<-** fread**(**input **=** "../input/test.tsv", header**=TRUE**, stringsAsFactors **=** **FALSE**, sep**=**'\t'**)**

The class of dataset read using fread() method is “data.table data.frame”. Thus, converting the class to data.frame.

train **<-** data.frame**(**train**)**

test **<-** data.frame**(**test**)**

1. **Structure of data –**

Obtain the class, number of observations and variables in the train and test datasets and datatype of each variable with few initial values for the datasets.

#Structure of dataset train

str**(**train**)**

#Structure of dataset test

str**(**test**)**

1. **Summary of target variable -**

Get the mean, median, min, max values for the price field which is our target variable.

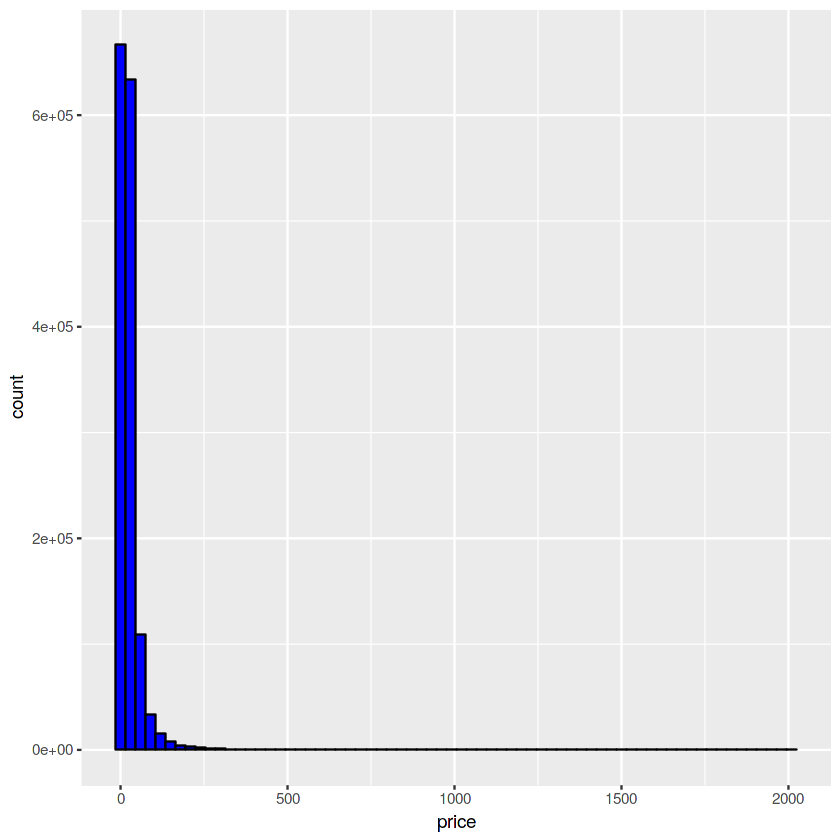
summary**(**train**$**price**)**

1. **Histogram of target variable –**

Plot the histogram for the target variable to get the skewness in the variable.

#Plot histogram of Price

ggplot**(**train, aes**(**x **=** price**))** **+** geom\_histogram**(**fill**=**"blue", colour **=** "black", binwidth **=** 30**)**

****

From the plot we can see that the price variable is highly skewed towards left. Thus, we transform the variable by taking the log.

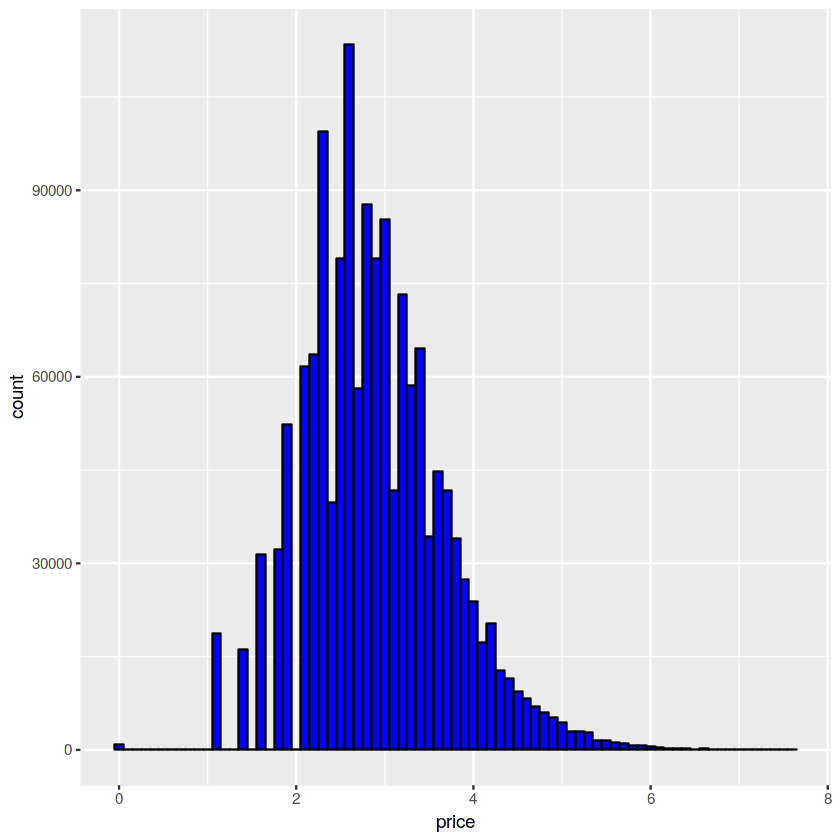
#Perform log transformation on price

train**$**price **<-** with**(**train, ifelse**(**price**>**0, log**(**price**)**, price**))**

Plot the transformed variable

#Plot histogram of transformed price

ggplot**(**train, aes**(**x **=** price**))** **+** geom\_histogram**(**fill **=** "blue", colour **=** "black", binwidth **=** 0.1**)**

****

1. **Combine train and test –**

Store transformed price variable in another variable. Remove price column from train data, and combine both train and the test data for performing combined data analysis.

#Combine train and test

price\_train **<-** train**[**,"price"**]**

train **<-** train**[**, **-**which**(**names**(**train**)** %in% c**(**"price"**))]**

Store the number of rows in train and test in some variabes, so that we can split back train and test into separate datasets for further processing.

train\_num\_rows **<-** dim**(**train**)[**1**]**

test\_num\_rows **<-** dim**(**test**)[**1**]**

colnames**(**train**)[**1**]** **<-** "id"

colnames**(**test**)[**1**]** **<-** "id"

Combine train and test datasets into train\_test

train\_test **<-** rbind**(**train, test**)**

1. **Missing value Analysis –**

Find the number of missing values in each of the columns of the datasets. The variables that are of character datatype may have an empty string of value. Thus, checking for is.na as well as empty string.

#Missing Values Analysis

apply**(**train\_test, 2, **function(**x**){**sum**(**is.na**(**x**)** **|** x **==** ""**)})**

1. **Plot factor variables –**

Plot all the variable which are of factor datatype.

#Plot brands and their frequency

train\_test**$**brand\_name**[**train\_test**$**brand\_name**==**"" **|** is.na**(**train\_test**$**brand\_name**)]** **<-** "Missing\_Brand"

#Not considering missing brand

top\_brands **<-** sort**(**table**(**train\_test**$**brand\_name**)**, decreasing **=** T**)[**2**:**20**]**

top\_brands **<-** data.frame**(**top\_brands**)**

str**(**top\_brands**)**

ggplot**(**top\_brands, aes**(**x **=** Var1, y **=** Freq**))** **+** geom\_bar**(**stat**=**"identity",fill**=** "DarkSlateBlue"**)** **+** xlab**(**"Brand"**)** **+** scale\_x\_discrete**(**labels **=** abbreviate**)**

#Separate out the three levels of category

train\_test**$**category\_name**[**train\_test**$**category\_name **==** "" **|** is.na**(**train\_test**$**category\_name**)]** **<-** "Missing1/Missing2/Missing3"

train\_test**$**level1\_category **<-** str\_split\_fixed**(**train\_test**$**category\_name, "/" ,3**)[**,1**]**

train\_test**$**level2\_category **<-** str\_split\_fixed**(**train\_test**$**category\_name, "/" ,3**)[**,2**]**

train\_test**$**level3\_category **<-** str\_split\_fixed**(**train\_test**$**category\_name, "/" ,3**)[**,3**]**

#Plot level1\_category and their frequencies

top\_l1\_cat **<-** sort**(**table**(**train\_test**$**level1\_category**)**, decreasing **=** T**)[**1**:**10**]**

top\_l1\_cat **<-** data.frame**(**top\_l1\_cat**)**

ggplot**(**top\_l1\_cat, aes**(**x **=** Var1, y **=** Freq**))** **+** geom\_bar**(**stat**=**"identity",fill**=** "DarkSlateBlue"**)** **+** xlab**(**"Top level1 category"**)**

#Plot level2\_category and their frequencies

top\_l2\_cat **<-** sort**(**table**(**train\_test**$**level2\_category**)**, decreasing **=** T**)[**1**:**10**]**

top\_l2\_cat **<-** data.frame**(**top\_l2\_cat**)**

ggplot**(**top\_l2\_cat, aes**(**x **=** Var1, y **=** Freq**))** **+** geom\_bar**(**stat**=**"identity",fill**=** "DarkSlateBlue"**)** **+** xlab**(**"Top level2 category"**)**

#Plot level3\_category and their frequencies

top\_l3\_cat **<-** sort**(**table**(**train\_test**$**level3\_category**)**, decreasing **=** T**)[**1**:**10**]**

top\_l3\_cat **<-** data.frame**(**top\_l3\_cat**)**

ggplot**(**top\_l3\_cat, aes**(**x **=** Var1, y **=** Freq**))** **+** geom\_bar**(**stat**=**"identity",fill**=** "DarkSlateBlue"**)** **+** xlab**(**"Top level3 category"**)**

1. **Data Preparation**

**Based on the understanding of the data, clean and prepare the data to make it machine learning ready.**

1. **Load the libraries –**

#Librarires

library('data.table') #For fread

library('tictoc')

library('stringr')

library('tm')

library('dummies')

library('xgboost')

library('sampling')

1. Load data -

#load data

train **<-** fread**(**input **=** "../input/train.tsv", header**=TRUE**, stringsAsFactors **=** **FALSE**, sep**=**'\t'**)**

test **<-** fread**(**input **=** "../input/test.tsv", header**=TRUE**, stringsAsFactors **=** **FALSE**, sep**=**'\t'**)**

train **<-** data.frame**(**train**)**

test **<-** data.frame**(**test**)**

1. Transform price column -

train**$**price **<-** with**(**train, ifelse**(**price**>**0, log**(**price**)**, price**))**

1. Take sample of data due to its large size.

stratas **<-** strata**(**train, c**(**"item\_condition\_id"**)**, size **=** c**(**100000,100000,100000,10000,1500**)**, method **=** "srswor"**)**

train **<-** getdata**(**train, stratas**)**

train **<-** train**[**,**-**which**(**names**(**train**)** %in% c**(**"ID\_unit","Prob","Stratum"**))]**

1. Combine train and test data

price\_train **<-** train**[**,"price"**]**

train **<-** train**[**, **-**which**(**names**(**train**)** %in% c**(**"price"**))]**

train\_num\_rows **<-** dim**(**train**)[**1**]**

test\_last\_id **<-** dim**(**test**)[**1**]**

colnames**(**train**)[**1**]** **<-** "id"

colnames**(**test**)[**1**]** **<-** "id"

train\_test **<-** rbind**(**train, test**)**

1. Missing value imputation –

train\_test**$**brand\_name**[**train\_test**$**brand\_name**==**"" **|** is.na**(**train\_test**$**brand\_name**)]** **<-** "Missing\_Brand"

train\_test**$**item\_description**[**train\_test**$**item\_description**==**"No description yet" **|** train\_test**$**item\_description**==**""**]** **<-** "NoDescYet"

1. Create Document term matrix for item\_description column

#DTM for item\_description

#Form corpus

corpus **<-** Corpus**(**VectorSource**(**train\_test**$**item\_description**))**

#Remove columns from train data

train\_test **<-** train\_test**[**, **-**which**(**names**(**train\_test**)** %in% c**(**"item\_description"**))]**

#str(train)

#Clean data

corpus **<-** tm\_map**(**corpus,tolower**)**

corpus **<-** tm\_map**(**corpus, removeNumbers**)**

corpus **<-** tm\_map**(**corpus, removePunctuation**)**

corpus **<-** tm\_map**(**corpus, removeWords, stopwords**(**'english'**))**

corpus **<-** tm\_map**(**corpus, stemDocument, "english"**)**

corpus **<-** tm\_map**(**corpus, stripWhitespace**)**

dtm **<-** DocumentTermMatrix**(**corpus**)**

dtm **<-** removeSparseTerms**(**dtm, 0.99**)**

train\_test **<-** data.frame**(**train\_test, as.matrix**(**dtm**))**

1. Create Document term matrix for item\_description column

#DTM for name

#Form corpus

name\_corpus **<-** Corpus**(**VectorSource**(**train\_test**$**name**))**

train\_test **<-** train\_test**[**, **-**which**(**names**(**train\_test**)** %in% c**(**"name"**))]**

name\_corpus **<-** tm\_map**(**name\_corpus, tolower**)**

name\_corpus **<-** tm\_map**(**name\_corpus, removeNumbers**)**

name\_corpus **<-** tm\_map**(**name\_corpus, removePunctuation**)**

name\_corpus **<-** tm\_map**(**name\_corpus, removeWords, stopwords**(**"english"**))**

name\_corpus **<-** tm\_map**(**name\_corpus, stripWhitespace**)**

dtm\_name **<-** DocumentTermMatrix**(**name\_corpus**)**

dtm\_name **<-** removeSparseTerms**(**dtm\_name, 0.99**)**

train\_test **<-** data.frame**(**train\_test, as.matrix**(**dtm\_name**))**

train\_test**$**item\_condition\_id **<-** as.factor**(**train\_test**$**item\_condition\_id**)**

train\_test**$**shipping **<-** as.factor**(**train\_test**$**shipping**)**

1. Separate category column into three levels -

#Separate out the three levels of category

train\_test**$**category\_name**[**train\_test**$**category\_name **==** "" **|** is.na**(**train\_test**$**category\_name**)]** **<-** "Missing1/Missing2/Missing3"

train\_test**$**level1\_category **<-** str\_split\_fixed**(**train\_test**$**category\_name, "/" ,3**)[**,1**]**

train\_test**$**level2\_category **<-** str\_split\_fixed**(**train\_test**$**category\_name, "/" ,3**)[**,2**]**

train\_test**$**level3\_category **<-** str\_split\_fixed**(**train\_test**$**category\_name, "/" ,3**)[**,3**]**

1. Convert all the non-numeric data to numeric –

This is done so that we can apply XGBoost algorithm on the dataset.

#Convert all data to numeric

#1. Convert brand name to numeric

#select top 20 brands based on frequency

top\_brands **<-** sort**(**table**(**train\_test**$**brand\_name**)**, decreasing **=** T**)[**1**:**21**]**

top\_brands **<-** data.frame**(**top\_brands**)**

top\_brands\_list **<-** top\_brands**$**Var1

train\_test**$**brand\_name **<-** as.factor**(**ifelse**(**train\_test**$**brand\_name %in% top\_brands\_list, train\_test**$**brand\_name, "other\_brand"**))**

Brands **<-** dummy**(**train\_test**$**brand\_name**)**

train\_test **<-** data.frame**(**train\_test, Brands**)**

#Convert level1\_category to numeric

train\_test**$**level1\_category **<-** as.factor**(**train\_test**$**level1\_category**)**

Level1\_category **<-** dummy**(**train\_test**$**level1\_category**)**

train\_test **<-** data.frame**(**train\_test, Level1\_category**)**

#Convert level2\_category to numeric

#Select top 20 level 2 categories based on frequency

top\_l2\_cat **<-** sort**(**table**(**train\_test**$**level2\_category**)**, decreasing **=** T**)[**1**:**21**]**

top\_l2\_cat **<-** data.frame**(**top\_l2\_cat**)**

top\_l2\_cat\_list **<-** top\_l2\_cat**$**Var1

train\_test**$**level2\_category **<-** as.factor**(**ifelse**(**train\_test**$**level2\_category %in% top\_l2\_cat\_list, train\_test**$**level2\_category, "other\_l2\_cat"**))**

Level2\_category **<-** dummy**(**train\_test**$**level2\_category**)**

train\_test **<-** data.frame**(**train\_test, Level2\_category**)**

#Convert level3\_category to numeric

#Select top 20 level 3 categories based on frequency

top\_l3\_cat **<-** sort**(**table**(**train\_test**$**level3\_category**)**, decreasing **=** T**)[**1**:**21**]**

top\_l3\_cat **<-** data.frame**(**top\_l3\_cat**)**

top\_l3\_cat\_list **<-** top\_l3\_cat**$**Var1

train\_test**$**level3\_category **<-** as.factor**(**ifelse**(**train\_test**$**level3\_category %in% top\_l3\_cat\_list, train\_test**$**level3\_category, "other\_l3\_cat"**))**

Level3\_category **<-** dummy**(**train\_test**$**level3\_category**)**

train\_test **<-** data.frame**(**train\_test, Level3\_category**)**

#Convert item\_condition\_id to one hot encoding

Item\_condition\_id **<-** dummy**(**train\_test**$**item\_condition\_id**)**

train\_test **<-** data.frame**(**train\_test, Item\_condition\_id**)**

#Remove columns not required

train\_test **<-** train\_test**[**, **-**which**(**names**(**train\_test**)** %in% c**(**"brand\_name", "level1\_category", "level2\_category", "level3\_category", "category\_name", "item\_condition\_id"**))]**

1. **Modelling**

**Get the train and the test data back from the combined train\_test data.**

#Split train and test again

train **<-** train\_test**[**1**:**train\_num\_rows,**]**

test **<-** train\_test**[**train\_num\_rows **+** 1**:**test\_last\_id,**]**

dim**(**train**)**

dim**(**test**)**

First convert the shipping column in test and train data to numeric. Use XGBoost model with linear regression. Using linear regression because the target variable is continuous, thus it is a prediction problem. Remove column id from the training data.

#Build model

train**$**shipping **<-** as.numeric**(**train**$**shipping**)**

model **<-** xgboost**(**data **=** as.matrix**(**train**[**,**-**c**(**1**)])**, label **=** price\_train, nrounds **=** 70, objective **=** "reg:linear"**)**

1. **Prediction**

Use the model obtained in the 4th step to predict the price of the products in the test data. Remove column id from the test data.

test**$**shipping **<-** as.numeric**(**test**$**shipping**)**

predicted\_price **<-** predict**(**model, as.matrix**(**test**[**,**-**c**(**1**)]))**

Write the results in a csv file.

#Write results into csv

predicted\_price **<-** data.frame**(**exp**(**predicted\_price**))**

csv\_output **<-** data.frame**(**test**$**id, predicted\_price**)**

colnames**(**csv\_output**)** **<-** c**(**"test\_id", "price"**)**

write.csv**(**csv\_output, "output.csv", row.names **=** F**)**