# Product Price Prediction
# Project Overview

**Business Objective**
Clothing has strong seasonal pricing trends and is heavily influenced by brand names, while electronics have fluctuating prices based on product specs.
Mercari, 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.
In this project, Mercari's challenging us to build an algorithm that automatically suggests the right product prices.

**Data Overview**
There are two files available. They are train.tsv and test.tsv
Both are tab separated files
The following are the data fields

1. **train_id or test_id** - the id of the listing
2. **name** - the title of the listing. Note that we have cleaned the data to remove text that looks like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]
3. **item_condition_id** - the condition of the items provided by the seller
4. **category_name** - category of the listing
5. **brand_name**
6. **price** - the price that the item was sold for. This is the target variable that you will predict. The unit is USD. This column doesn't exist in test.tsv since that is what you will predict.
7. **shipping** - 1 if shipping fee is paid by seller and 0 by buyer
8. **item_description** - the full description of the item. Note that we have cleaned the data to remove text that look like prices (e.g. $20) to avoid leakage. These removed prices are represented as [rm]

**Source**
https://www.kaggle.com/c/mercari-price-suggestion-challenge/overview/description

**Aim**
To predict the price of the product using the given description and other information

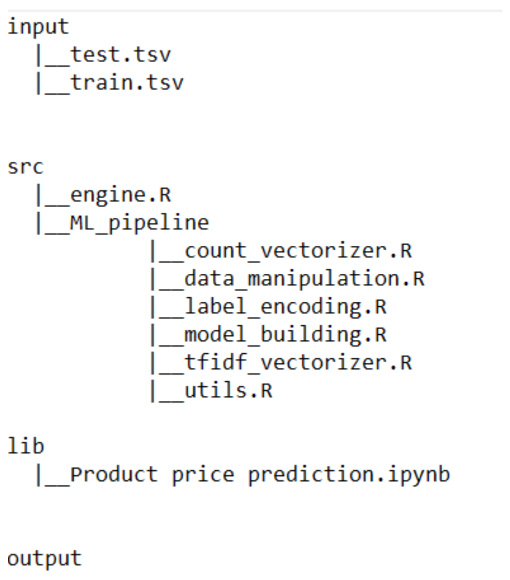
**Tech Stack**
- ➔ Language used : R
- ➔ Packages used : superml, textstem, neuralnet, gbm, quantenda, and so on..
- ➔ UI support : R Shiny Dashboard

**Approach**

1. **Exploratory Data Analysis**
   Exploratory data analysis is the process of analysing the dataset to understand its characteristics. In this step, we perform the following.
   a. Univariate analysis - Analysis of a single variable
   b. Bivariate analysis - Analysis of relationship between two variable
2. **Data cleaning / Pre-processing (outlier/missing values/categorical)**
   Machine learning algorithms for regression can understand the input only in the form of numbers and hence it is highly essential to convert the non - numeric data that we have to numeric data by providing them labels.
   a. Label Encoding
3. **Missing value treatment**
   This step involves the process of filling the missing values in appropriate ways so that the data is not lost.
4. **Feature Engineering**
   a. CountVectorizer
   b. TFIDF for text data
5. **Modelling**
   Various regression algorithms are applied on the dataset and the model that suits best for the dataset is selected. The models that we apply for this dataset are
   a. Random forest
   b. SVM
   c. Evaluation
   d. Neural networks
   e. Evaluation

**Modular code overview**

The ipython notebook is modularized into different functions so that the user can use those functions instantly whenever needed. The modularized code folder is structured in the following way.



```
input
  |__test.tsv
  |__train.tsv


src
  |__engine.R
  |__ML_pipeline
           |__count_vectorizer.R
           |__data_manipulation.R
           |__label_encoding.R
           |__model_building.R
           |__tfidf_vectorizer.R
           |__utils.R

lib
  |__Product price prediction.ipynb


output
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input
2. src
3. output
4. lib

1. The input folder contains all the data that we have for analysis. In our case, it will contain two tsv files which are
    a. train.tsv
    b. test.tsv
2. The src folder is the heart of the project. This folder contains all the modularized code for all the above steps in a modularized manner. It further contains the following.
    a. ML_pipeline
    b. engine.R

The ML_pipeline is a folder that contains all the functions put into different R files which are appropriately named. These python functions are then called inside the engine.R file

3. The output folder contains all the models that we trained for this data saved as .rds files. These models can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
4. The lib folder is a reference folder. It contains the original notebook that we saw in the videos.

**Project Takeaways**

1. Performing basic EDA
2. Advanced EDA ideas
3. Checking and handling null values

4. Performing slicing and making function for converting variables into categorical types
5. Understanding and using TFIDF for analyzing textual data
6. Understanding and using Count Vectorizer for analyzing textual data
7. Applying LabelBinarizer for textual data
8. Building and training a Random forest model
9. Building and training a SVM model
10. Building and training a Evaluation model
11. Building and training a Neural networks model
12. Drawing predictions from the trained model
13. Evaluating and fine tuning the model
14. Building API for the model using flask