# PROJECT REPORT

# Mercari Price Suggestion

**Description:-**

It can be hard to know how much something’s really worth. Small details can mean big differences in pricing. For example, one of these sweaters cost 335 and the other cost 9.99. Can you guess which one’s which?

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, 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 competition, Mercari’s challenging you to build an algorithm that automatically suggests the right product prices. You’ll be provided user-inputted text descriptions of their products, including details like product category name, brand name, and item condition.

**Data fields:-**

train.tsv, test.tsv

The files consist of a list of product listings. These files are tab-delimited.

train\_id or test\_id - the id of the listing

name - the title of the listing. 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]

item\_condition\_id - the condition of the items provided by the seller

category\_name - category of the listing

brand\_name

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.

Shipping - 1 if shipping fee is paid by seller and 0 by buyer

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]

**Road map:-**

Road map which I followed :

1) Understand the data

2) Perform statistical analysis

3) Make some hypothesis and validate them

4) Fill in missing values (data preprocessing)

5) Make some visualizations ( Univariate and bivariate analysis)

6) Analyzing the text columns

7) Text preprocessing

8) Make some visualizations on text columns

9) Count vectorizer and TFIDF

10) Building ML

1. **Understand the data**:

So we have the business statement and variables given to us, From the business statement we get to know that, this is a prediction problem and the target variable is price and 6 independent variables are given to us.

4 variables are text related , 2 variables are numerical data

I took a sample of 10000 rows

1. **Perform statistical analysis**:

I did perform statistical analysis, on numerical data I performed mean, sd, percentiles and total

On the categorical variables I saw there count and there distribution.

Price is right skewed and mean in around 26.9

On the categorical variables we find that in item\_categorical\_id has 5 categories

1 is the highest and 5 is the lowest.

In shipping 0(not paid by the seller) is more then the paid

1. **Make some hypothesis and validate them**

From the hypothesis I found the following

* From the analysis we get to know that we have a relationship between the price and category name and price and brand
* so my hypothesis is right there is an relationship between price and item\_condition\_id
* distribution of price is right skewed

1. **Filling in the missing values and dummy variables:**

* Missing values there filled for categorical name and brand name filled by ‘None’
* Dummy variables where created to shipping and condition id

1. **Make some visualizations ( Univariate and bivariate analysis)**

* Univariate analysis visualization was done on the condition id , shipping and price
* Bi variant analysis was also done

1. **Analysis of text and visualizations**

Used count vectorizer and TFIDF

Visualization is also been shown

1. **Model building**:-

Best was (ridge regression)

**Random forest** :- **Random forests** or **random decision forests**[[1]](https://en.wikipedia.org/wiki/Random_forest#cite_note-ho1995-1)[[2]](https://en.wikipedia.org/wiki/Random_forest#cite_note-ho1998-2) are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks, that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting  to their training set.

Metrics: RMSLR

I got an accuracy of about 0.62

**Gradient boosting**: **Gradient boosting** is a ML technique for regression and classification problems, which produces a prediction model in the form of an ensenble of weak prediction models, typically decision tree.

For like 50 estimates I got around 0.65

**XGBOOST**: its type of gradient boosting but is scalable and reduces the over fitting and performs better then GBM

Prediction value: 0.59

**Ridge regression** :

* It shrinks the parameters, therefore it is mostly used to prevent multicollinearity.
* It reduces the model complexity by coefficient shrinkage.

Prediction value 0.56