**Retail Price Optimisation**

**Problem statement:**

Managing prices by the retail industry is the key to profit in business. It is essential to identify a reasonable price range and make adjustments for the prices of products accordingly.

The user need to build best model takes in sales data, product Characteristics, on the structure data, textual information, and pricing rules and several inputs. The model suggest price for a product in real-time by considering thousands of relationships within a product.

**Approach:**

1. Collected data from kaggle.

2. Imported necessary libraries and packages into google colab.

3. Loaded data into data frame.

4. Done descriptive analysis on data and found various columns and their datatypes.

The price of items are right skewed, vast majority of the items priced at 10–20. However, the most expensive item priced at 2009. So we will make log-transformation on the price.

Over 54% of items shipping fee were paid by the buyers.

5. We found that average price is 22.03 if seller pays shipping, average price is 30.14 if buyer pays shipping.

6. We found that the average price is higher when buyer pays shipping.

7. There are different average prices for each item condition id. I decide to use all the features to build our model.

8.Found the missing values and filled it with the values “missing”.

9. Changed category\_name , brand\_name and item\_condition\_id to categorical dtype.

10. Deleted rows with price 0.

11. Applied countvector() on category name to convert a collection of text documents to a vector of term/token counts

12. Applied tdifvectorizer() on item\_description to map the most frequent words to feature indices and hence compute a word occurrence frequency (sparse) matrix.

13.applied labelbinarizer() on brand\_name to accepts Categorical data as input and returns an Numpy array.

14. Applied getdummies() on item\_condition\_id and shipping then converted it into csr matrix.

15.stackted sequence of X\_dummies,X\_description, X\_brand, X\_category, X\_name horizontally using hstack() and converted it into csr matrix.

16.Removed features with document frequency less than one.

17. Splitted data into train and test and fitted/trained the lightgbm model. Light gbm is framework that is used to make predictions using trees.

18.Got 0.6783877979337455 as RMSE(Root Mean Square Error).