Mercari Price Prediction Taskforce.

- Data exploration / Data cleaning
- Feature engg / selection
- Performance metric (RMSE ?)
- Model evalu^{n} & selection / optimization
- Conclusion / report doc

← Already did part →

To visualize text data —plot→ word clouds
- Sort data acc. to prices (low → high)
- Divide data into 4 equal parts
- 1st quarter → products with lowest prices
- 4th Quarter → products with highest prices.

Observt^{n}: Sellers try to put good words (never used, brand new etc) to sell the products quickly
commonly found in all quarters.

— % rows with No-descript^{n} yet.

* Boxplot to check if products are impacted when description is given or not.
  ↳ not much diff (median at both occasions is almost same).

wht abt 75th percentile values in both? —

Features selected for modelling:
1) brand_name
2) brand_name_given
6) subcategory_1
7) subcategory_2

3) item_description_id
4) shipping
5) main_category
8) item_description

Note: While applying ONE-HOT, apply it to test data with respect to train data to avoid data lekage

To represent item_description feature:
- Lets apply Bag of words to test in simple. (similar to One-hot encoding)
  ↳ builds vocabulary of unique words in our dataset & associate unique column to each word in vocab.
- Each sentence is rep as list with length = # distinct words in vocabulary
  @ each col. in list, mark how many times given word appears in sentence. (ignores order of words).

Evaluation metric:
* RMSE (Root Mean Squared Error) is used for regression tasks.

becz of skew in target col. (price), RMSLE (log error) would be relevant.

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

RMSLE

ϵ → RMSLE score
n → #total observations in public/private data set.
p_i → your prediction of price
a_i → actual price

Establishing a baseline:
To compare our results against.
For regression prblms → mean of price in train set.
Baseline error → _____ .

Model selection / Hyper-param tuning

- Try diff algos & see which works best
  - Linear Regression
  - Support vector Regression (?)
  - Decision Tree Regression (?)
  - Logistic Regression
- Above models can be optimized by tuning their **Hyperparameters.** (eg: # trees in random forest, # neighbours in K-nearest)

Model parameters :→ which model learns during training (eg: weights in Linear Regression)

→ control balance bet **Underfitting** & **Overfitting**

Underfitting: when model is not complex enough to learn map from features → target (High **bias**)
corrected by making model more complex

Overfitting: sort of memorizes training data. (High variance)
corrected by limiting complexity of model tho' **Regularization**

⊛ There's no universal correct hyperparameters, every ML problm will be diff.

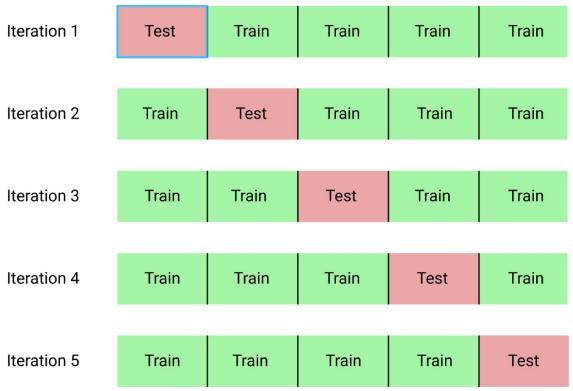
Hyper-parameter Tuning technique:

→ Random search with K-fold cross validation.

↳ defines a grid, then randomly sample diff combinations & perform K-fold CV.

K-fold CV: Given dataset is split into K. no of sections / folds, where each fold is used as testing set at some point of time.

Test error from all iters are averaged for that specific hyper-parameter combination.



eg: 5-fold cross-validation.
(one-fold used for test, rest all for training.