Ensemble Learning Project

Effect of online product description on its sales

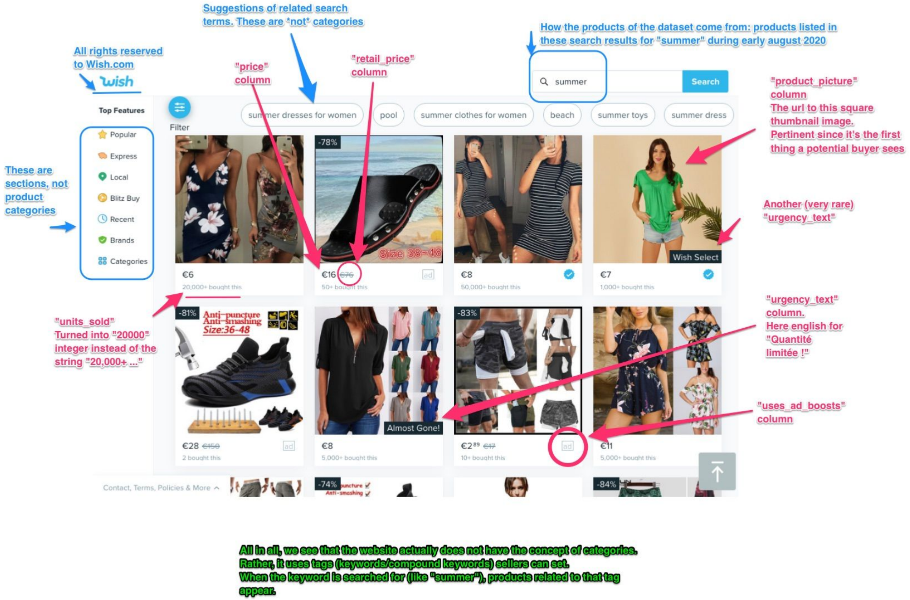
Group #4
Romain COLLEDANI | Theo COSTES | Jeremie FERON | Antoine GALLIER | Alexandra GAUTRON

Outline

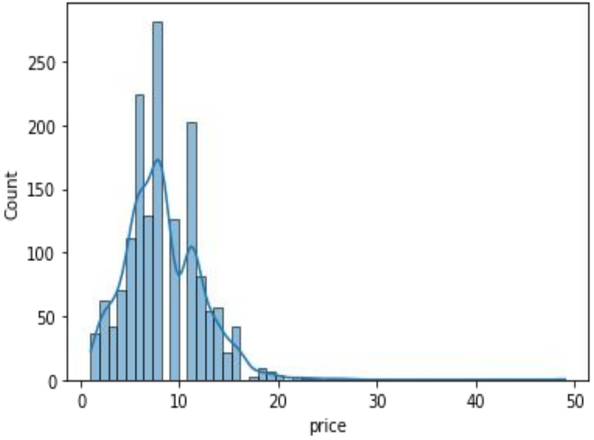
1. Introduction
2. Data Analysis
3. Data Pre-processing
4. Data Analysis after Pre-processing
5. Models
6. Results & Model Selection
7. Interpretability
8. Conclusion

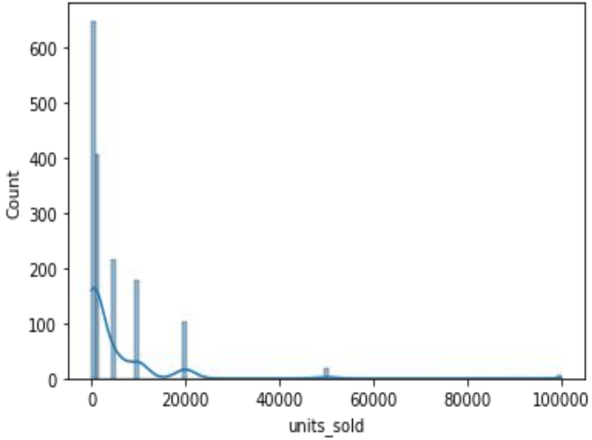
Introduction

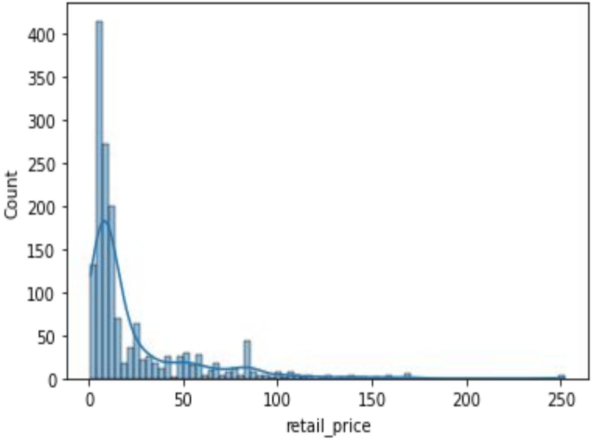
- Data from the eCommerce website **Wish**
- Information on **1500+ products** : price, color, title, merchant, country, ...
- Analysis on what influences the sales of a product online (**units_sold**)
- Several **Ensemble Methods** developed and compared to select the most appropriate

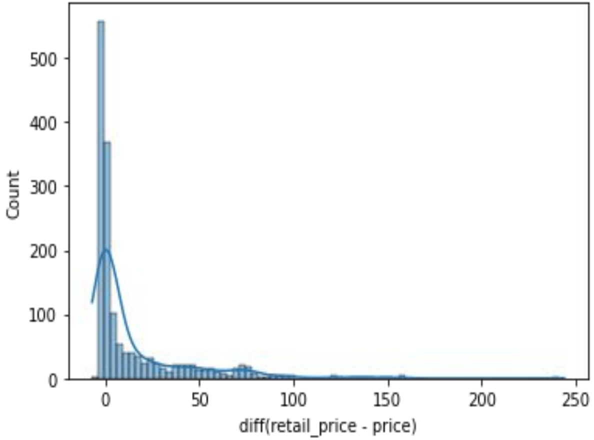


Data Analysis

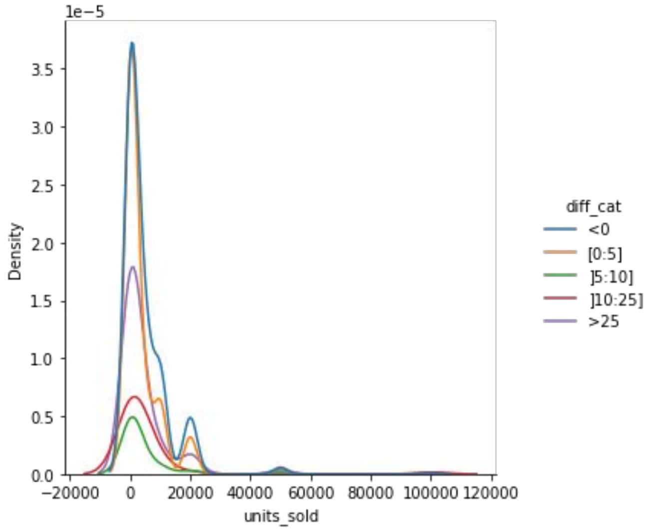


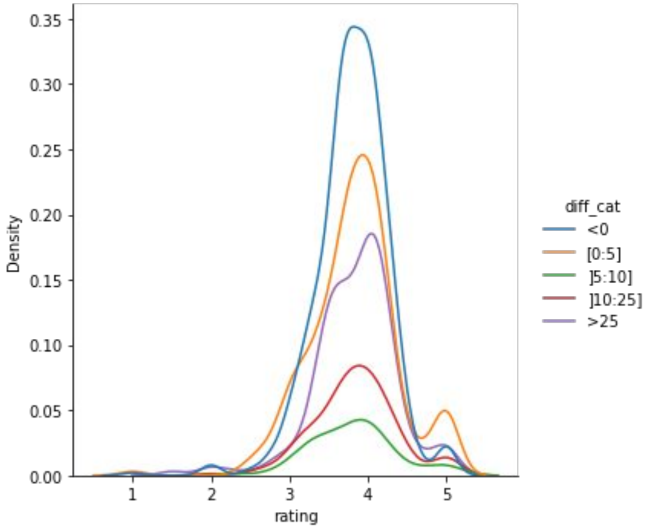






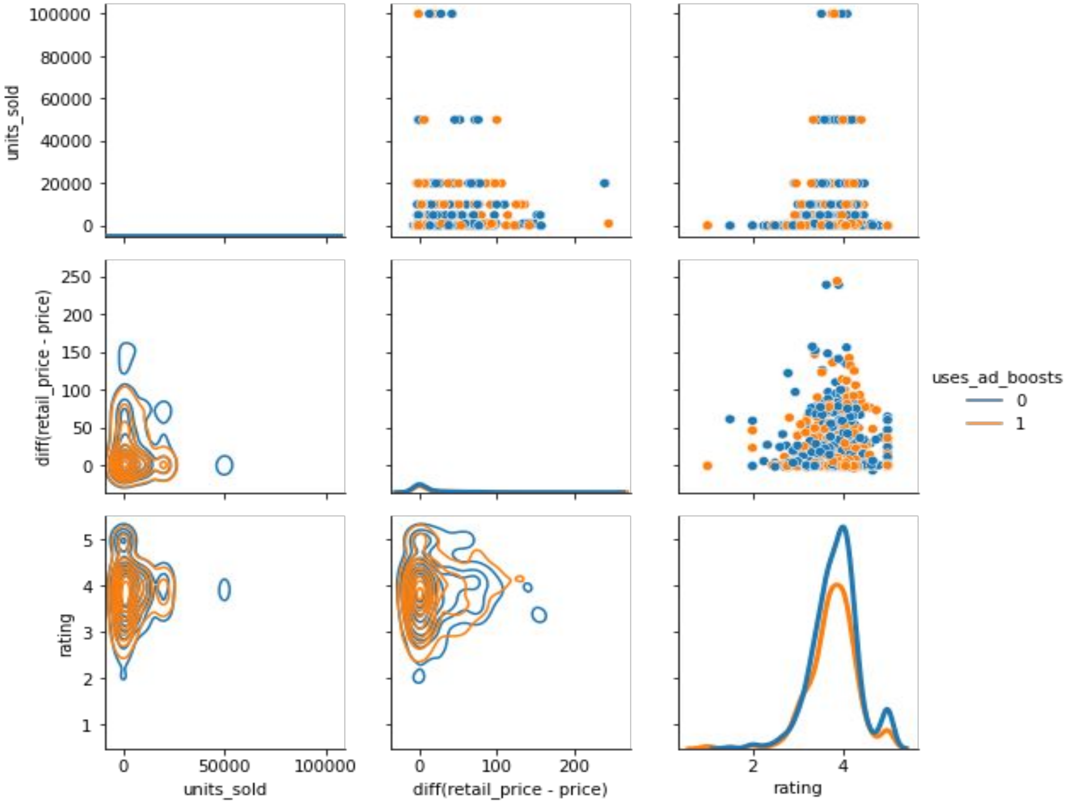
Data Analysis





- We noticed that the **difference** between the **retail price** and the **selling price** has an **influence** on the distribution of units sold. As the larger the difference, the more equally distributed the units sold is.
- **Same** happened with respect to the **rating**.

Data Analysis

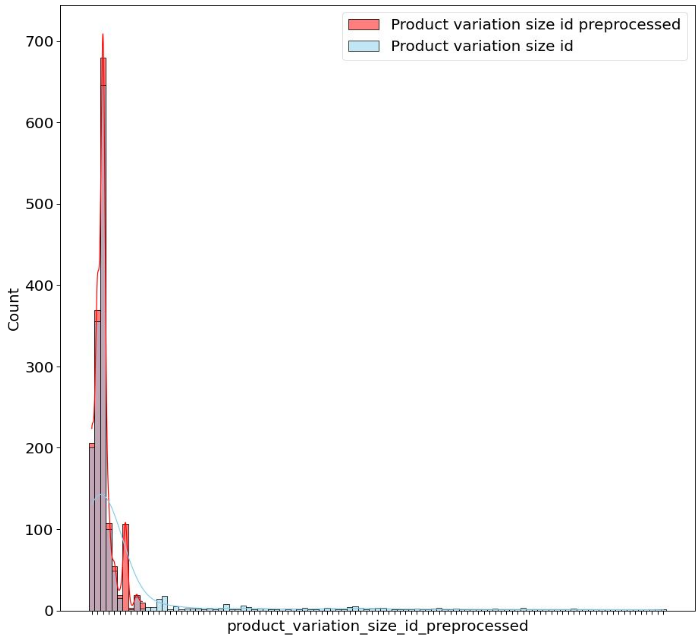


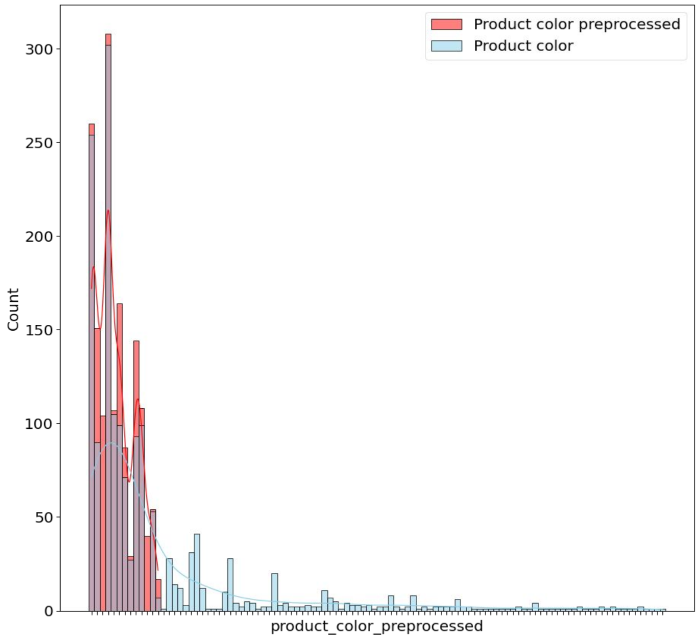
- It’s very **difficult** to **detect** graphically an **impact** of variables on the units sold. → we need to use some ensemble learning methods

Data Pre-processing

- **Creating new more relevant features:**
    - **Textual** data : TF-IDF to estimate the impact of each product description its turnover
    - **Difference** between merchant's and product's **ratings**
- Deleting the **useless** features (e.g., **URL**s or features with only **one** possible **value**)
- **Encoding categorical** features
- **Removing redundant** information
- Dealing with **missing values** :
    - Replace missing rating counts by 0
    - Creating dummy features
- Transforming the **target variable** into a **categorical** feature (classification task with 6 classes)

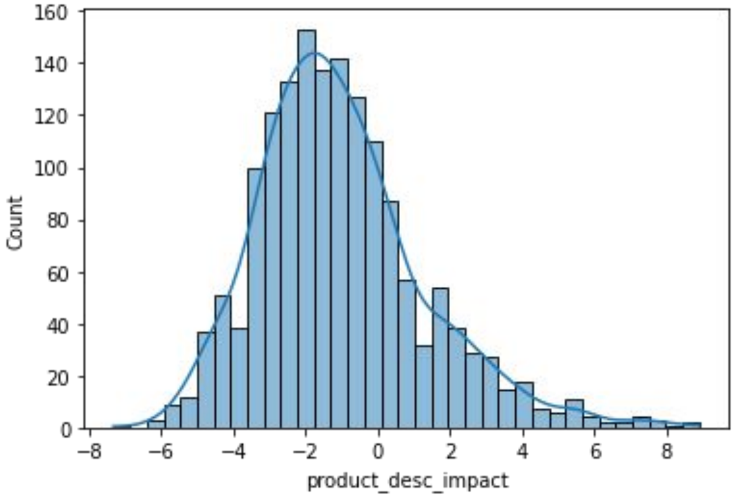
Data Analysis after Pre-processing





- The pre-processing part allows us to **reduce** the **sparsity** of some important categorical variables.

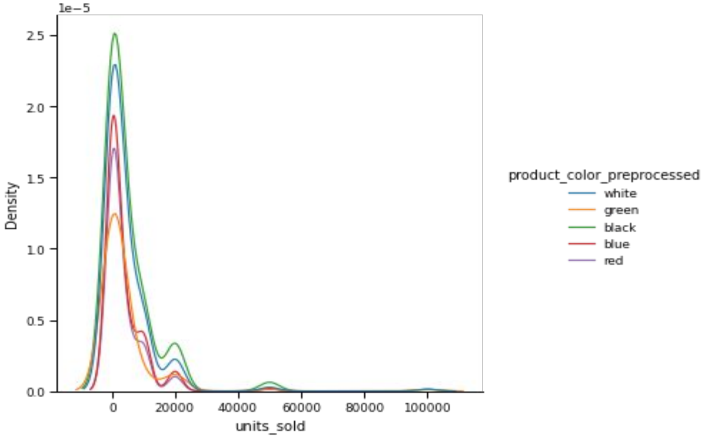
Data Analysis after Pre-processing

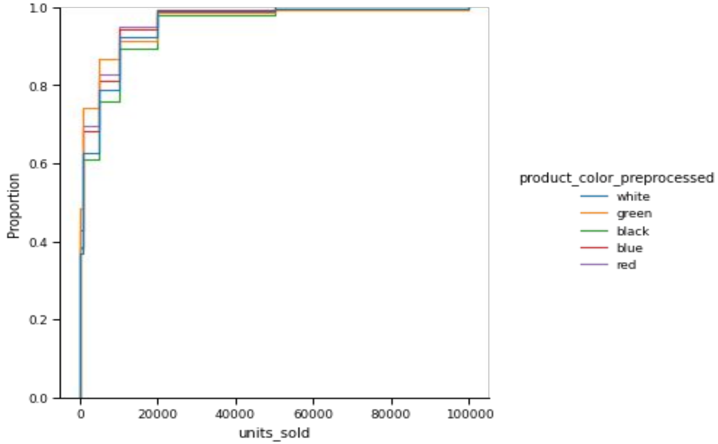


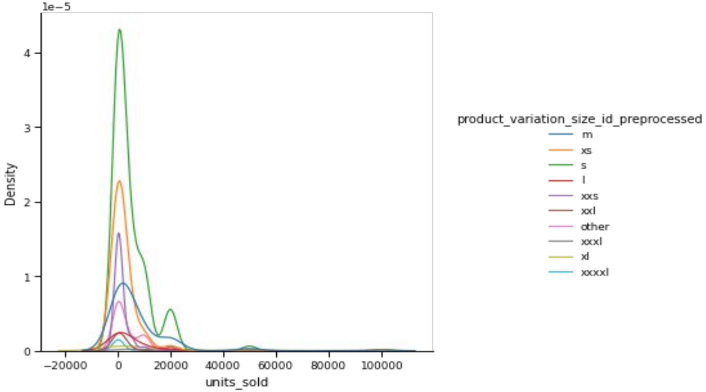
- Our new column gives us a **quantitative information** about the positive/negative **impact** of a **comment** in the **units sold**.
  → The **distribution** is **centered in -2**. We may notice that **some comments** can have a really **powerful positive impact** of sells (>5) or **negative.**

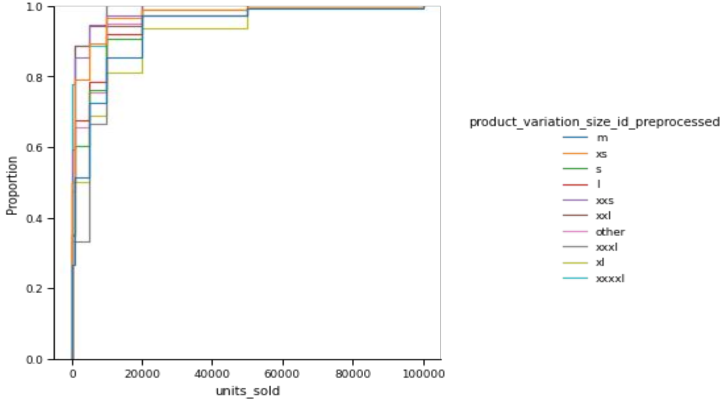
  → We may want to analyse the **behaviour** of this very **impactful comments.**

Data Analysis after Pre-processing









→ Very **difficult** to **interpret**, we have to use ensemble models

Models

- **Different models** to predict the target variable (units_sold) :
  - Decision Trees,
  - Bagging,
  - Random forests,
  - Extremely Randomized Trees,
  - Gradient Boosted Trees,
  - AdaBoost,
  - XGBoost,
  - Stacking (Random Forest + Extra Trees).

- **Hyperparameters tuning** through a (5-fold) cross-validated Grid Search on the **train** set

- One common **metric** to evaluate the performance on the **test** set : the **weighted F1 Score**

$$Precision = \frac{T_p}{T_p + F_p} \qquad Recall = \frac{T_p}{T_p + T_n} \qquad F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

"weighted" : Average of binary metrics in which each class' score is weighted by its presence in the true data sample

The closer to **1** the F1 Score, the better the model

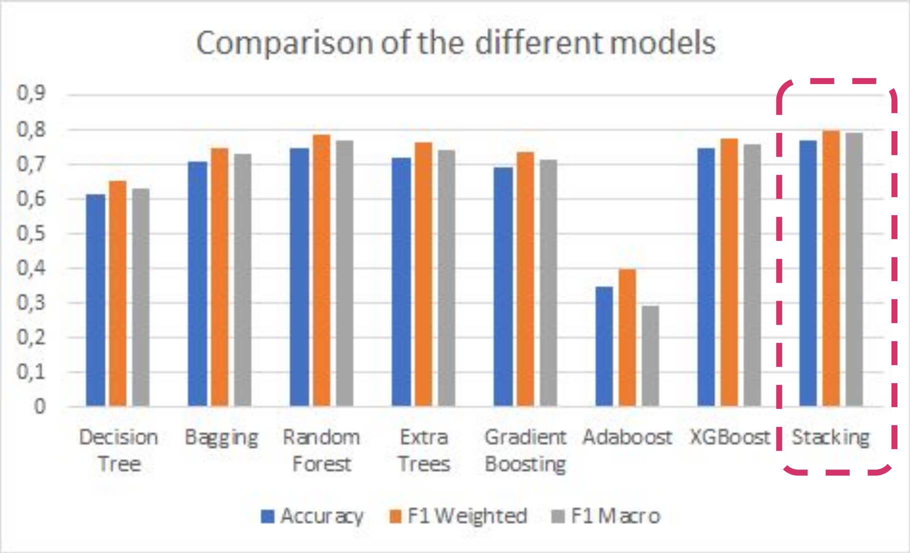
Results & Model Selection

	Decision Tree	Bagging	Random Forest	Extra Trees
Hyperparameters	max_depth: 5 min_samples_leaf: 2	max_features : 0.4 max_samples : 0.8	n_estimators : 500 max_depth : 15 min_samples_leaf : 1	n_estimators : 300 max_depth : 25 min_samples_leaf : 1

	Gradient Boosting	Adaboost	XGBoost	Stacking
Hyperparameters	learning_rate: 0.1 max_depth: 10 min_samples_leaf: 3 n_estimators: 100	learning_rate: 0.1 n_estimators: 400	learning_rate: 0.1 max_depth : 4 min_child_weight: 1 n_estimators: 100	Random Forest : best parameters Extra trees : best parameters

- All **other parameters** were set to **default** and **unchanged** across models (e.g., split criterion - Gini).

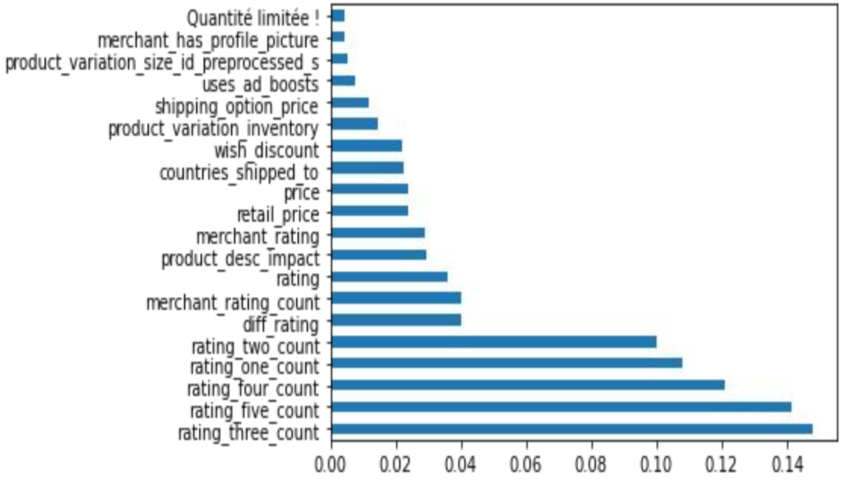
Results & Model Selection



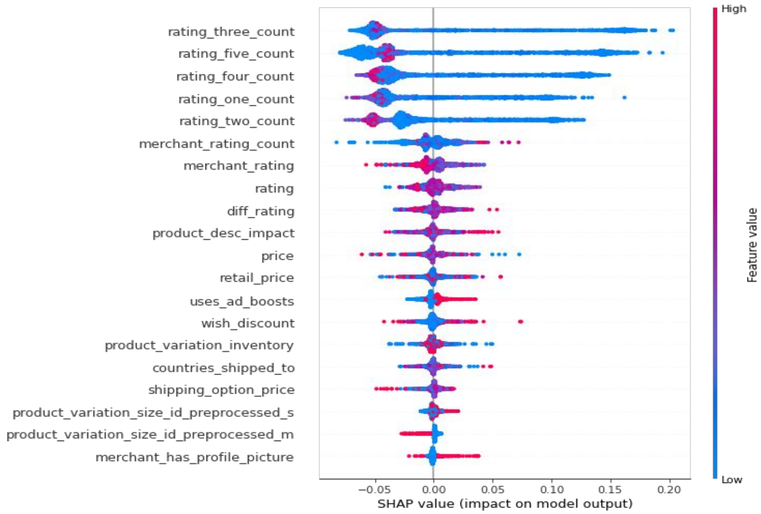
- **Results** across the different models tested were in the **same range** ([0.6,0.8]) except for Adaboost.
- **Stacking** was the **best** performing model of all the indicators (Random Forest-XG Boost very close).

Interpretability

- Most **impactful features** for the final best model (computed from RF and Extra RF):



Impurity-based feature importances



Shapley Values = weighted average of marginal contributions

- Rating counts seemed to capture much of the information which is an expected behaviour as it makes the seller more credible. Interestingly enough it had more impact than the average rating.

Conclusion

- Importance of **data pre-processing** and **feature engineering**
  - Hand crafted metrics like diff_rating, diff_price contributed to improve the results.
  - Interestingly, expressing the difference between 2 attributes (like price and retail_price) through a new feature effectively added information and allowed the algorithms to make better decisions.
  - Some **features** are more **important** than others: e.g., ratings, product_description_impact.

- **Stacking** is the most appropriate approach on this data set for the predictive task (**XG Boost** and **Random Forest** very close)
  - Limitations: There is no feature.importance directly available as stacking is a combination of RF and Extra RF. We have to average the feature importance of those models to get the feature importance of stacking.

- Ideas for further **improvement**:
  - Interpretability:
    - Impurity based feature importance is biased towards high cardinality values. We could use partial permutations and/ or growing unbiased trees to determine categorical variables importance.
    - Some of our features are very correlated with each other which undermines interpretability through permutation (e.g Rating counts). Grouping those features or using different techniques for interpretability could improve results.
    - Extract new features, for example out of product images.

Thank you !