**Data Exploration Findings**

1. Peak into data

We start with 32 features with ratings being the target feature.

2. Using a heat correlation map:

Columns price and shipping option price has a correlation of 0.89. Decisions: if correlation was higher, I would consider removing but 0.89 is still too low in my opinion and needs further consideration.

Columns rating count and units sold also have a correlation of 0.89. Decisions: if correlation was higher, I would consider removing but 0.89 is still too low in my opinion and needs further consideration. It also makes sense that these two are positively correlated.

Has urgency banner and urgency text have a correlation of 1 (determined after some preprocessing). Decisions: I first fixed the columns and missing values so they represent a binary feature and then re did the heat map to find that these two had a correlation of 1 and should therefore be removed.

Chart, scatter chart

Description automatically generated

3. Checking for missing values

Product color, product variation size id, has urgency banner, urgency text, origin country, merchant name, merchant profile picture are the only columns with missing values. Decisions: required further look into features to determine how to deal with missing values.

Checking for uni-value columns (columns with the same values for each row or composed MAINLY (95% +) of one value. (also double checked with excel filter)

* Currency buyer’s only value is EUR, theme and crawl\_month only have one value as well. Decisions: these features are removed.
* Shipping\_option\_name only has 3 categories however they are mainly China or US very disproportionately. Decisions: these features are removed.
* Shipping\_is\_express almost all the rows had a value of 0 and only a few 1s. Decisions: these features are removed.

4. Checking feature (and target variable) distribution

Checking the distribution of ratings to see what is considered a good rating. Seems that most ratings are approximately 4 and a good rating is anything above 4.3 let’s say. The two graphs represent the distribution of the target variable (rating). It seems that it is a binomial distribution with majority of ratings at 4. Decisions: I can try a data sampling method to deal with the slightly unbalanced data. It seems that a big proportion of rating is 4 so that class has the most amount of data oversampling (since we don’t have too much data) could be used to see if performance improves.

A picture containing histogram

Description automatically generatedChart

Description automatically generated

Checking distribution of numeric features (following are positively skewed)

|  |  |  |
| --- | --- | --- |
| **Feature** | **Mean** | **Median** |
| retail\_price | 24 | 10 |
| units\_sold | 4519 | 1000 |
| rating\_count | 916.0 | 143.5 |
| merchant\_rating\_count | 26784 | 8225 |

The above features are all positively skewed as they have a mean higher than their median. Decisions: to improve performance we could try cube/cube square to sort of improve the distribution of the feature.

5. checking for significant relationships between features/features and features/target

Wanted to see if there is a difference between price and retail\_price. And I also wanted to see if that difference has an effect of ratings. First, I set 4 as a threshold for a successful rating then compared difference in price and retail price with product rating. As you can see there is no different between high and low rated products in terms of price vs retail price. Decisions: Since there is no significant correlation, these two features could be removed if needed to improve model performance

A picture containing shape

Description automatically generated

I wanted to see the relationship between ad\_boosts and product rating. There seems to be no significant relationship between using ad boosts and ratings. Decisions: could be removed if needed to improve model performance

Chart, bar chart

Description automatically generated

I wanted to see the relationship between merchant profile and merchant rating with product rating.There seems to be no significant association between merchant rating and rating of the product as seen in the violinplot (right). However, there is a relationship between merchant profile and product ratings (left). It seems that presence of merchant profile picture is significantly associated with higher ratings.

Chart, bar chart

Description automatically generatedChart

Description automatically generated

**Data cleaning and pre-processing**

6. Editing ‘string’ columns to binary

Both ‘urgency\_text’ and ‘has\_urgency\_banner’ contained missing values and one had tex and needed to be pre-processed. Decisions: both were fixed in terms of missing values and changed to binary e.g. 1 if text or banner was present and 0 if not.

7. Unify name variations

The features ‘product\_variation\_size\_id’ and ‘product\_color’ both name variations refereeing to the same thing e.g. ‘light blue’ ‘blue’ ‘Blue’ ‘Dark blue’ ‘bluee’ are all variations of blue. Decisions: this was fixes by grouping major variations under one category e.g. ‘blue’ and categorizing minor variations as ‘OTHER’. This was done for both of these features. There were a lot of variations in the color category in terms of spelling and abbreviations of same colors. Decisions: I grouped all variation into the respective color. Missing values or indistinguishable color names were categorized under OTHER. I also made a distribution of colors (left) just to see what color is the most popular. It seems that the most popular colors are black and white. Most popular sized are small and extra small.

Chart, bar chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

8. Combining minority groups due to highly skewed distribution

For the feature origin\_country it is mostly composed of US, China and minor others. Decisions: I preprocessed the data so that I keep CN and US and group the rest into others along with those with missing values.

Chart, bar chart, histogram

Description automatically generatedChart

Description automatically generated

Units\_sold has few values below then first majority are above . Decisions: i combined anything below 10 as 10.

A picture containing table

Description automatically generated

9. Duplicate row removal

Duplicate rows were found. Decisions: they were removed going from 1094 to 1081.

10. Column removal

Duplicate rows Different features (columns) were removed and added back to observe the performance of the model. The list shown in the accompanying Jupyter code consists of features that consistently produced better results. Decisions: the table below shows all the features and the final decision made for the feature after playing around with model performance (Table 1).



**Normalizing / Scaling / Encoding features / feature engineering**

11. Encoding non int features

Features such as ‘product\_color’ , ‘product\_variation\_size\_id’ and ‘origin\_country’ were ended using the get\_dummies pandas function. Also the feature ‘tags’ were modified to count how many tags were in each record (using commas) and that number was saved as ‘tag\_count’ and added as a feature and ‘tags’ was removed.

12. Min/Max normalization

All features except for the target variables ‘rating’ were normalized using the MinMaxScaler function from sklearn. Models were trained with both normalized and un-normalized features to check for performance.

13. Feature Engineering

After running the sklearn feature importance function with my random forest model I saw that most consistently important features were ratings and rating counts. I wanted to create a feature that distinguishes high rating with low rate counts and vice versa. I created this feature for rating\_count and merchant\_rating\_count. Both features were used in the prediction model and evaluated using feature importance function.

Text

Description automatically generated

**Final data preparations**

14. train / test split

I created 4 sets of X and y. first is X\_no\_split , y\_no\_split which is used for cross validation later on. Second is X\_not\_nor , y\_not\_nor which contains un-normalized data. Third is X\_nor , y\_nor which contains normalized data. Fourth and last (which only takes the training set) us called X\_res , y\_res which represents oversampled training data (running this cell was optional to observe change in performance).I mainly created these to make it easier to run models multiple times without having to keep manually changing variables to observe change in performance. I wanted to see the effect of normalization, over-sampling, cross validation on model performance.

**Model evaluation**

*General feature patterns:*

Wanted I observed the performance of all models using by removing and adding various features. Some general patterns I observed were slightly better performance for all models when color and size attributes were removed. I further confirmed these when I ran feature evaluation using the sklearn random forest feature evaluation and saw that features such as color, size, shipping is express, badge\_local\_product had the lowest priority (significance). These columns were then dropped. The remaining are 17 features used for modeling in order of importance in the following figure.

**Text

Description automatically generated**

*Normalized vs un-normalized:*

|  |  |  |
| --- | --- | --- |
| Model | Mean F1 score | |
| Normalized | Un-normalized |
| Decision Tree | 0.647 | 0.647 |
| Random Forest | 0.704 | 0.704 |
| Adaboost | 0.703 | 0.704 |
| Neural net | 0.24 | 0.193 |
| Ensemble learning | 0.708 | 0.708 |

There was no significance improve in performance except for a slight improvement for the neural net and ensemble learning. Since no negative effects were observed I will use normalized values for prediction.

*Cross validated vs not cross validated:*

|  |  |  |
| --- | --- | --- |
| Model | Mean F1 score | |
| Cross validated | Not cross validated |
| Decision Tree | 0.495 | 0.647 |
| Random Forest | 0.555 | 0.704 |
| Adaboost | 0.555 | 0.704 |
| Neural Net | 0.194 | 0.193 |
| Ensemble Learning | 0.543 | 0.708 |

\*\* Adaboost took a very long time to run for cross validated (like… very long)

Since scores were lowered after cross validating results it shows that my model might not be as good as I thought right now. For a more realistic gasp of model performance I will be using cross validation to double check my model performance to make sure a good performance value isn’t just by chance.

*Over-sampling vs not over-sampling:*

|  |  |  |
| --- | --- | --- |
| Model | Mean F1 score | |
| Over-sampled | Not over-sampled |
| Decision Tree | 0.615 | 0.647 |
| Random Forest | 0.668 | 0.704 |
| Adaboost | 0.677 | 0.704 |
| Neural net | 0.265 | 0.193 |
| Ensemble learning | 0.650 | 0.708 |

And I decided to try over sampling the data because we had an un-even number of records per class, there were many records for rating 4 and not as much for the other ratings. However, over-sampling data did not seem to increase model performance therefore will be not be implemented.

**Decision Tree**

* Gini performs better than entropy
* Splitter=’random’ improved performance
* Increases max-depth increases performance but has the risk of overfitting
* Model performs best when max\_features includes all features. Performs worse if you decrease it.

**Random Forest**

* Increasing number of estimators improved model performance
* Gini performs better than entropy
* Setting bootstrap to False increases model performance. Also boostrap is better to use than out of bag sampling.
* Setting warm\_start to True slightly increases model perfomance

**Adaboost**

* Lowering learning\_rate to 0.01 increases model performance.
* Lowering n\_estimators slightly increases performance

The table below shows difference pre-processing techniques used for each model variance. These models were also ran with varying set of features, new features were included and removed to observe performance.

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Model | | |
| XG Boost | Random Forest | Ensemble Learning |
| 1 | Not normalized  Not oversampled | Not normalized  Not oversampled | Not normalized  Not oversampled |
| 2 | Normalized  Not oversampled | Normalized  Not oversampled | Normalized  Not oversampled |
| 3 | Not normalized  Oversampled | Not normalized  Oversampled | Not normalized  Oversampled |
| 4 | Normalized  Oversampled | Normalized  Oversampled | Normalized  Oversampled |
| 5 | Normalized  Undersampled | Normalized  Undersampled | Normalized  Undersampled |

The highest score achieved uses training dataset was 0.7698 (on Kaggle) was using ensemble learning which contained the algorithms random forest, XGboost, Adaboost. The list of features used to achieve this is shown below.

**Text

Description automatically generated**

The dataset was normalized and cross validated and was not oversampled, in fact I noticed no significant difference with an oversampled dataset. In general ensemble learning and random forest and adaboost performed the best. XGboost and decision tree were next. Models such as neural net (MLP), KNN, and SVM did not perform well at all and were not further considered.