Machine Learning Engineer Nanodegree

Capstone Project

Yingying Hu
September 29, 2050

I. Definition

Project Overview

"How well is a product likely to sell?" is a crucial question for every retailer. Before the coming of eCommerce, brick-and-mortar businesses can only rely on intuition and experience when forecasting customer's demand on a product. Nowadays, more and more people choose to shop online because of the convenience and product varieties. At the same time, business owners can gain more customer data by tracking their activity on the shopping website. Analyzing and understanding what data suggests product sales performance will help businesses to optimize their stocks. Furthermore, a machine learning algorithm can bring it to the next level by predicting product sales and make decisions on product stock for business owners.

Wish is a popular American online e-commerce platform that facilitates transactions between sellers and buyers. A Kaggle user, Jeffrey Mvutu Mabilama, built a dataset which contains information about products and their merchants. This project will analyze the dataset and build machine learning models to predict the number of units sold for each product.

Problem Statement

The goal of this project is to create ML models to predict the product sales. Tasks including:

1. Explore and preprocess the “Sales of summer clothes in E-commerce Wish” dataset.
2. Select important features for predicting product units sold
3. Choose 3 ML models, including Random Forest, XGBoost and Neural Network
4. Train the models and check their performance by calculating Root Mean Square Error (RMSE)
5. Select a model from the 3 basic models and do hyperparameter tuning on SageMaker
6. Test the final model’s performance via Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), and compare them with the benchmark model

Metrics

Since the goal is to predict product sales, the model's label is numerical. The model's evaluation metrics will be Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). MAE and RMSE are two common metrics used to measure model performance for continuous labels. Since the benchmark model uses the MAE, it will be used for comparison purpose.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \qquad \text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|\hat{y}_i - y_i\right|$$

II. Analysis

Data Exploration

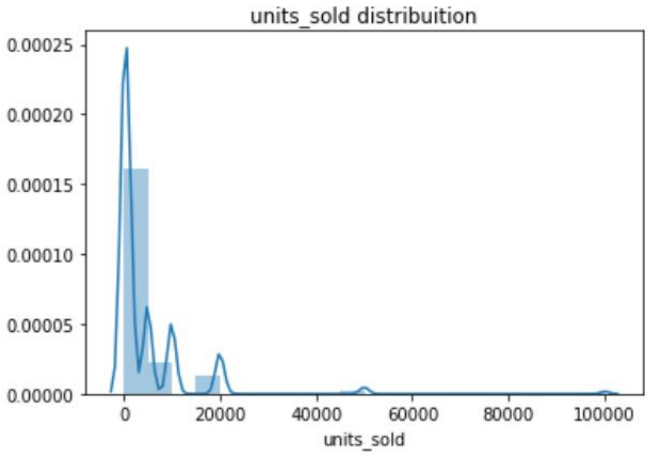
The "Sales of summer clothes in E-commerce Wish" dataset has three tables. In this project, only two of them are used. The tables are downloaded from Kaggle as csv files. The main table is called "summer-products-with-rating-and-performance_2020-08", which contains all summer related products available for sale, as of July 2020. The second table is "unique-categories-sorted-by-count". It has the occurrence of each tag keyword that appears in the "tag" column of the main table.

The original products table contains 1573 rows and 43 columns.

	title	title_orig	price	retail_price	currency_buyer	units_sold	uses_ad_boosts	rating	rating_count	rating_five_count	...	merchant_ratin
0	2020 Summer Vintage Flamingo Print Pajamas Se...	2020 Summer Vintage Flamingo Print Pajamas Se...	16.00	14	EUR	100	0	3.76	54	26.0	...	568
1	SSHOUSE Summer Casual Sleeveless Soirée Party ...	Women's Casual Summer Sleeveless Sexy Mini Dress	8.00	22	EUR	20000	1	3.45	6135	2269.0	...	17752

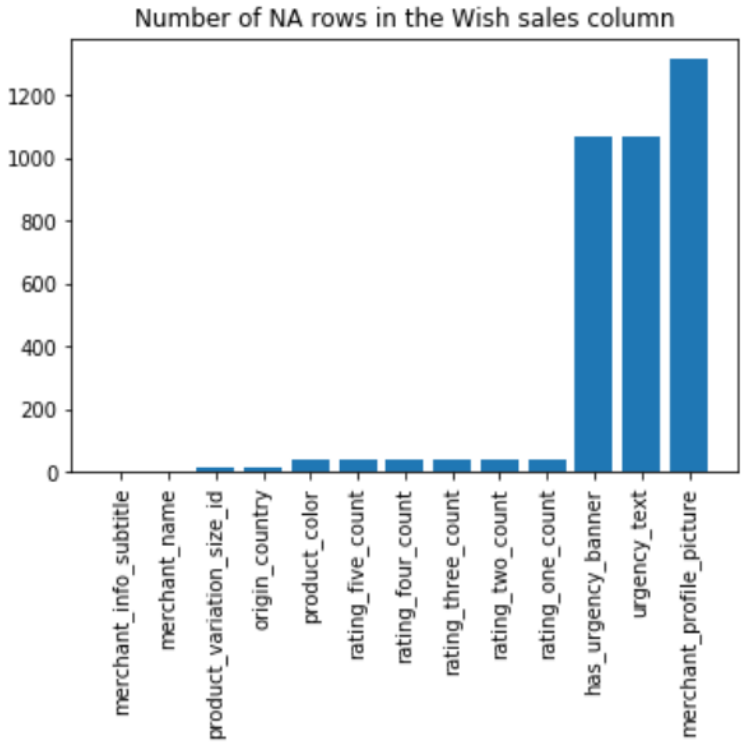
Part of the Product Table

Each entry represents a product. The dependent variable is `units_sold`. By observing its histogram and estimated PDF, the label is right-skewed. A log transformation will need to be applied on the `units_sold`



units_sold distribution

During data validation, 34 rows are found to be duplicates. Also, there are missing values in the dataset.



The majority columns in the main table are categorical. Some are binary, and some have a few hundred unique values. In addition, there are ten columns which includes irrelevant information, such as the product id, product url, the merchant's title, and the table's metadata. Since they are inapplicable for the goal of this project, they are dropped during the exploration.

The tag keyword table contains 2620 unique categories (entries), and it has no missing value.

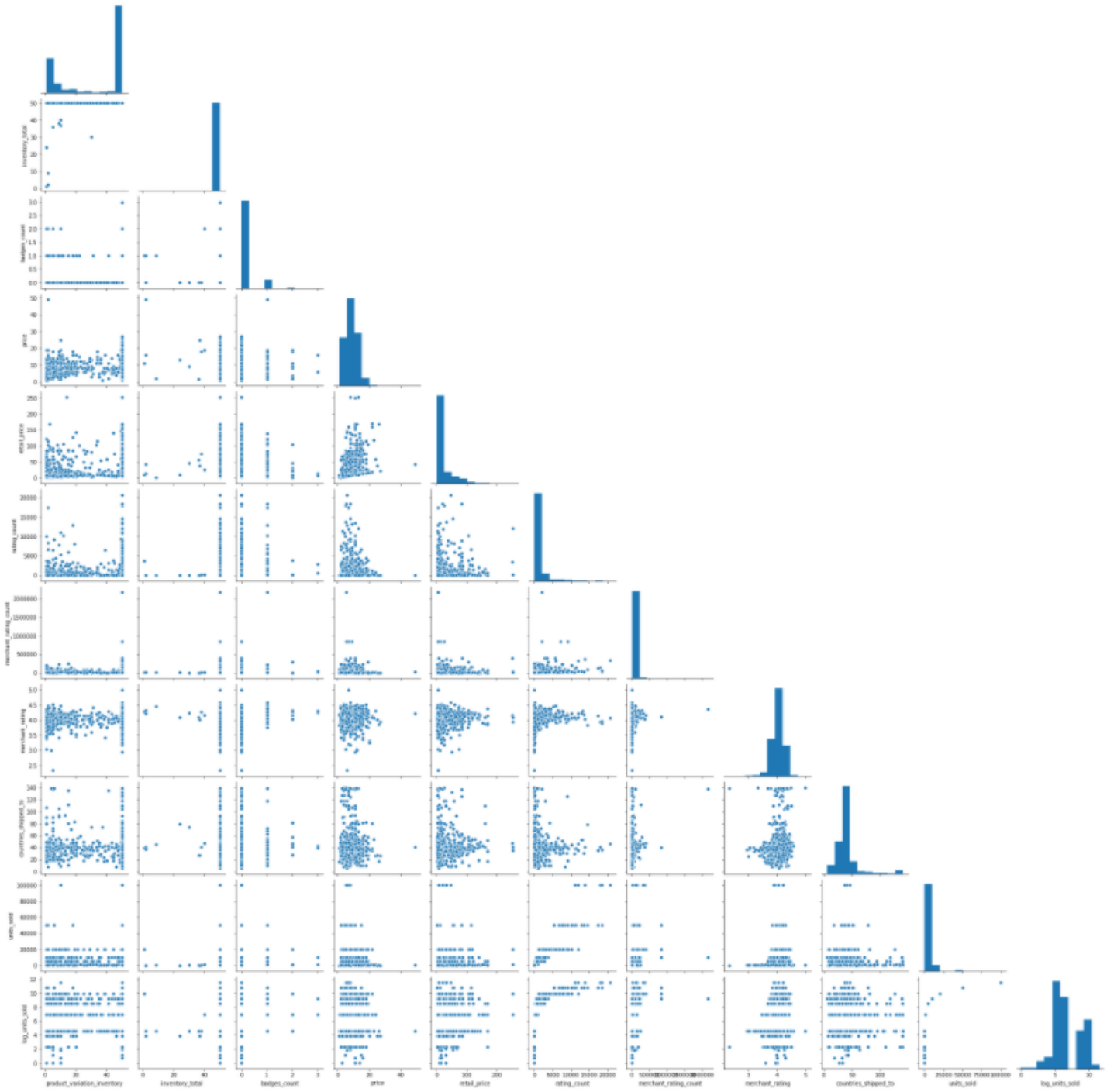
	count	keyword
0	1321	Summer
1	1315	Women's Fashion
2	1082	Fashion
3	961	Women
4	905	Casual

Top 4 rows in the Tag Keyword Table

The top 20 most frequent tags make up 41% of the total. In addition, some keywords express similar meaning. For example, the “Women’s Fashion”, “Fashion”, and “Women” are closely relevant.

Exploratory Visualization

Pair plot is a straightforward way to visualize the relationships between each numerical variable.

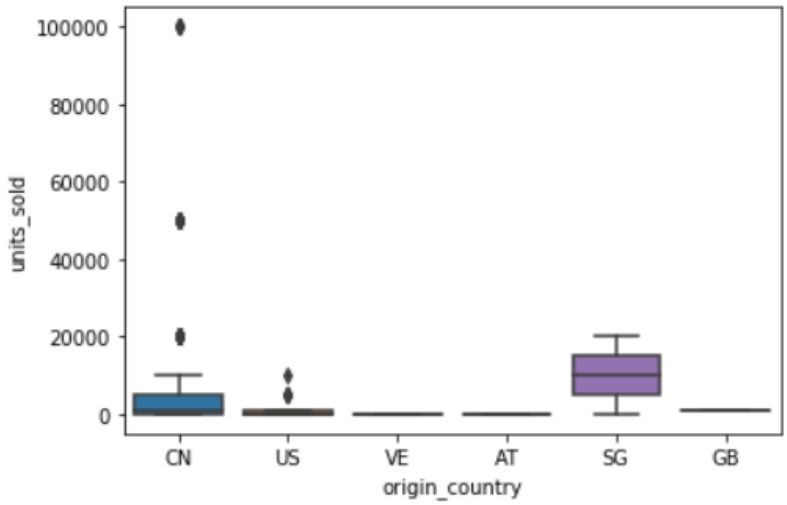


Pair plot for numerical features in the Product Table

The pair plot is made up by several bar plots and scatterplots. Bar plots on the diagonal shows the distribution of each numerical feature, and scatterplots on the lower left are used to investigate the possible relationship between two variables. The pair plot shows that some columns, like `inventory_total`, are severely unbalanced, which means the majority falls on a single value. These

features might not be useful. In addition, the scatterplot between price and log_units_sold (units_sold after taking log transformation) shows that cheaper products tend to have more sales. There is no obvious relationship between countries_shipped_to and the label.

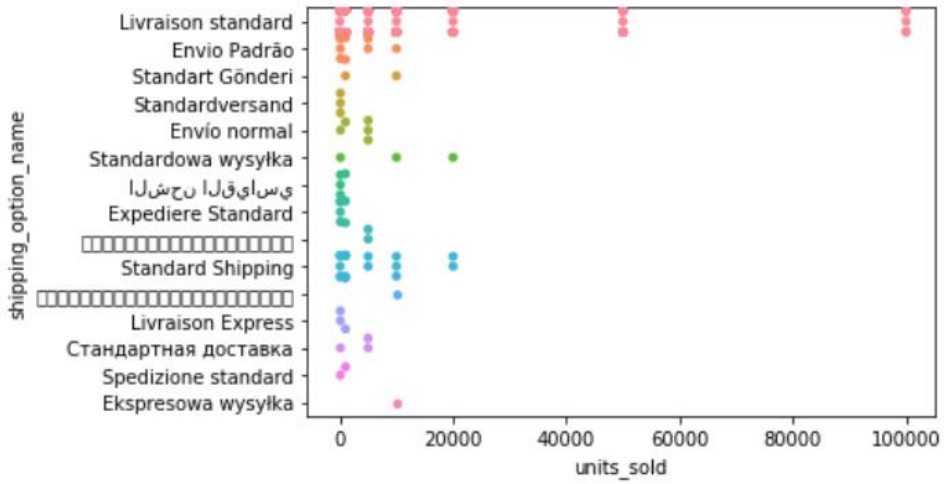
To visualize the relationship between a categorical variable and a continuous variable, this project uses the box plot.



Box Plot for units_sold VS. origin_country

The box plot above is for units_sold and origin_country. origin_country shows where the product is made. By observing the box plot, one can get a general sense of the units_sold distribution on each of the category in origin_counrty. Products with extremely large number (20000+) of sales are made in China, which is consistent with common sense. There are not many products are from VE, US, and GB.

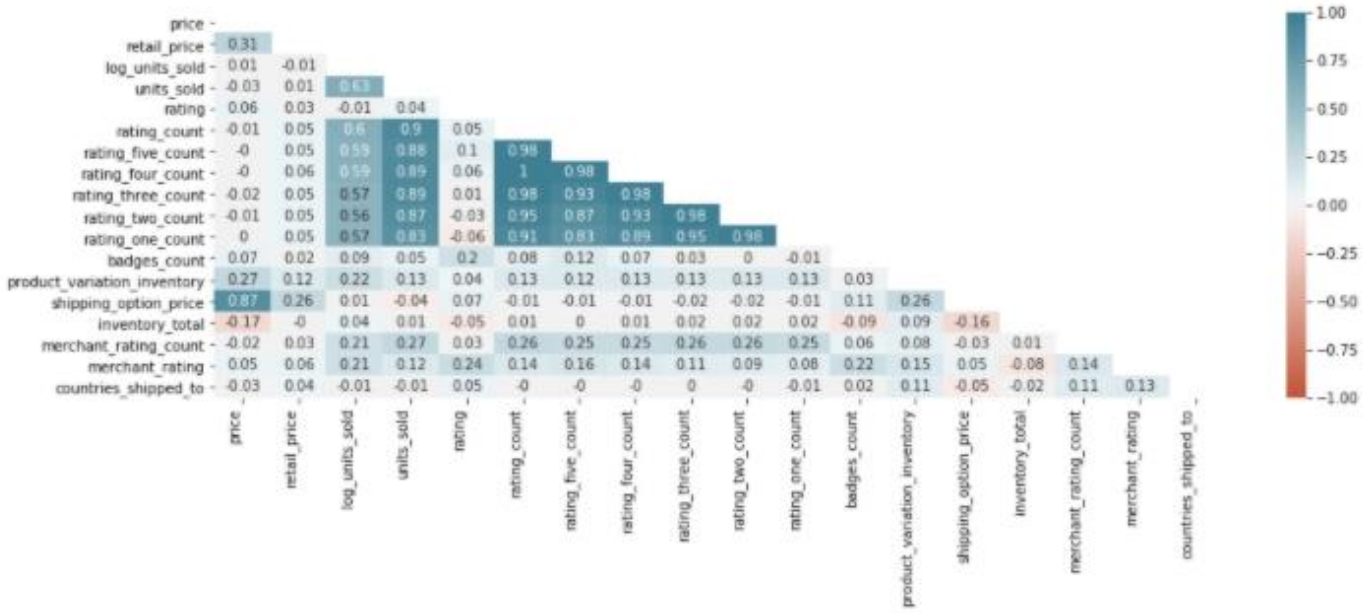
An alternative for plotting a categorical variable and a continuous variable is a categorical scatterplot.



Swarm Plot between units_sold and shipping_option_name

The above swarm plot shows units_sold and shipping_option_name. The different option name is labelled by different color. Livraison standard covers all the shipping for products with huge (40000+) sales.

To further check the relationship between numerical value. Their correlation matrix is calculated and shown in a heat map.



Correlation Matrix Heat Map

From the heat map, one can easily observe which two features are correlated, either positively or negatively. The color bar on the right side indicates what the color represents. Dark red means the correlation coefficient is -1 (negatively correlated), dark blue means 1 (positively correlated), and white means there is no correlation. For example, the square between shipping_option_price and price is dark blue, and it's correlation coefficient is 0.87.

Algorithms and Techniques

The goal of this project is to predict the units sold for each product, which is a regression problem. The benchmark model is a linear regression model, so this project tries to build advanced ML models to check if they can outperform the linear regression. It starts with building a Random Forest, XGBoost, and Neural Network, which are popular ML models for predicting the continuous label.

Benchmark

The benchmark model is a linear regression with combined L1 and L2 priors as regularizer. This model is shown in a public notebook shared by a Kaggle user. The model's MAE is 2045.58

III. Methodology

Data Preprocessing

Columns with missing values are handled one by one. It is notable that 3 columns (merchant_profile_picture , urgency_text, and has_urgency_banner) have 1000+ null values. The missing value is meaningful. For example, merchant_profile_picture has value 1 for seller has the profile picture. The null value in merchant_profile_picture indicates that the merchant does not have a profile picture, so the missing value is filled with 0. Similar solution is applied to the urgency_text, has_urgency_banner and all the rating_N_count columns. Other columns with much smaller number of null values are all categorical, and their missing value are replaced with a string value, “other”. The product_color is an interesting feature. It has more than a hundred unique values, so it will not be practical to apply one-hot encoding on all the values. The color white and black makes up about 35% of the whole. To transform this variable, the other colors are replaced with “other” before applying one-hot encoding. Same as origin_country where “China” accounts for ~95%. Hence, the other countries are replaced with “other”. Later, after making sure the dataset has no missing entries, all categorical variables are transformed via one-hot-encoding.

To utilize the tag keyword table, the top 20 tags are selected and used as new features. The top 20 tags cover about 41% of the total tags. The approach is to convert the tags column which is in the product table into 20 categorical columns by comparing the top 20 category keyword with the tags column. If a product has the tag that is in the top 20 tag, return a "Yes" (1), else return "No" (0). As mentioned before, some tags express similar meanings, which means they could be correlated. The next step is to handle features which are highly correlated.

For feature selection, the project starts with computing the correlation matrix for all the numerical features. Since correlation can only show if a feature is linearly related with the label, a correlation coefficient around 0 does not mean that there is no other type of relationship there. Hence, features with small correlation coefficient (around 0) are checked via scatterplot to make sure there is no-linear relationship. To check the association between each categorical feature, the project calculates Cramér's V. Features with Cramér's V greater than 0.6 are dropped. Then, the Point Biserial correlation coefficient between categorical variables and the numerical label is used to select the relevant nominal features.

In addition, the dependent variable, units_sold, is right-skewed. To make it less skewed, a log transformation is applied, and the new column will be used as the label for the model.

After data cleaning, transformation and feature engineering, the final table has 24 columns, including the dependent variable.

The last step before building the model is to split the dataset into training, validation, and testing set. Since the features are in different range, the project transforms feature by scaling each to a range of 0

and 1. The scaling is fit only on the training set, then the parameters are applied on scaling the validation and testing set.

Implementation

After data cleaning, feature engineering, transformation, scaling, and dataset splitting, the training table is ready for the model. For modelling, this project starts with training three ML models: Random Forest, XGBoost and Neural Network.

For the Random Forest model, this project uses the RandForestRegressor in the sklearn library. The model uses 20 trees, and the criterion is MSE which is same as the intended metric.

For XGBoost, there is a xgboost library, and all parameters simply uses the default values.

The Neural Network model is built via the MLPRegressor in the sklearn. The Neural Network has 2 hidden layers with 100 and 300 nodes on each.

After fitting the models, a 5-fold cross validation is used to check the estimator performance and if it is overfitting. The result table is:

Random Forest	`Scores: [0.74716593 0.68297109 0.68868995 0.73966999 0.70063254]`
	`Mean: 0.711825899665639`
	`Standard deviation: 0.026523100723128953`

XGBoost	`Scores: [0.74245291 0.67270865 0.7498293 0.75267925 0.72631802]`
	`Mean: 0.7287976270134355`
	`Standard deviation: 0.029500005977797412`

Neural Network	`Scores: [0.95516538 0.84125619 0.9774098 0.9324119 0.86176173]`
	`Mean: 0.9136010005381238`
	`Standard deviation: 0.05305486496409064`

The metric used by the cross validation is RMSE, smaller scores indicate the model performance is better. Hence, the random forest seems to be the best performed model.

Refinement

The result of initial implementation shows that the Random Forest outperforms the XGBoost model. This is contrary to the command sense as XGBoost usually tend to have better performance than Random Forest. One possible reason is that this XGBoost simply uses the default parameters. To verify the idea and improve XGBoost model's accuracy, the hyperparameter tuning is applied.

To utilize the hyperparameter tuning functionality on AWS SageMaker, the project switch from the xgboost library to the xgboost image on SageMaker. Also, the training, validation and testing set are uploaded to the S3 bucket. The hyperparameter includes :

- max_depth (Maximum depth of a tree)
- eta (learning rate)
- min_child_weight (Minimum sum of instance weight needed in a child
- subsample (Subsample ratio of the training instances)
- gamma (Minimum loss reduction required to make a further partition on a leaf node of the tree).

The validation metric used by the tuner is RMSE.

IV. Results

Model Evaluation and Validation

After hyperparameter tuning on AWS SageMaker, the best performed model is retrieved. The model's parameters are:

Best training job hyperparameters

Name	Type	Value
_tuning_objective_metric	FreeText	validation:rmse
early_stopping_rounds	FreeText	10
eta	Continuous	0.13597923064649742
gamma	Continuous	3.323488051231378
max_depth	Integer	12
min_child_weight	Integer	2
num_round	FreeText	200
objective	FreeText	reg:squarederror
subsample	Continuous	0.5493475284034675

To evaluate the model, its prediction on the test set data is compared with the true label. The model's RMSE is 0.65, which is better than the previous RandomForest and the first XGBoost without tuning.

The XGBoost model should be robust. Boosted Tree methods are fairly robust to outliers in the input features since the base learners are tree splits.

Justification

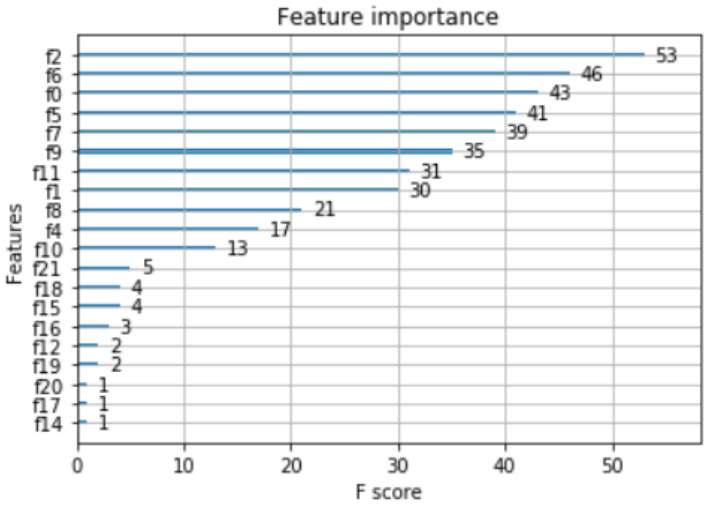
The next step is to compare the model's performance with the benchmark model, the MAE is used since the benchmark model the XGBoost MAE is about 1350, which is better than the benchmark model (2045.58).

V. Conclusion

Free-Form Visualization

The last step is to check the input features importance. The features importance score is necessary for analyzing which features are truly relevant for a product sale.

The final XGBoost is given 22 input features, 2 of them, however, are not used by the model. The two features are `badges_count` and `merchant_has_profile_picture`. To visualize other input features' importance, the F score for each feature is calculated and drawn in a plot.




```
{'f2': 'rating_count',
 'f6': 'merchant_rating',
 'f9': 'rating_three_count',
 'f7': 'rating_five_count',
 'f0': 'price',
 'f5': 'merchant_rating_count',
 'f10': 'rating_two_count',
 'f11': 'rating_one_count',
 'f8': 'rating_four_count',
 'f1': 'rating',
 'f14': 'size_m',
 'f16': 'size_s',
 'f4': 'product_variation_inventory',
 'f21': 'tag_tank',
 'f15': 'size_other',
 'f19': "tag_women's fashion",
 'f17': 'size_xs',
 'f18': 'tag_summer',
 'f20': 'tag_sexy',
 'f12': 'badge_product_quality'}
```

Corresponding Features Name

The top 5 important features are rating_count, merchant_rating, price, merchant_rating_count, and rating_five_count.
The top 5 least important features are size_m, size_xs, tag_sexy, tag_women's fashion and badge_product_quality.
The feature importance suggests that if a merchant wants to improve the products sales, the key is to build and maintain a good customer feedback.

Reflection

This project is worked on AWS SageMaker. The completed workflow includes:

1. Set up a git repository and a notebook instance on the AWS SageMaker platform
2. Download the data files from Kaggle and upload to a notebook instance
3. Explore, visualize, and clean the table, including handling missing values and duplicated entries
4. Feature engineering, such as check the correlation on numerical variables and perform one-hot encoding on the categorical features
5. Apply log transformation on the label column, units_sold, to make the data less skewed
6. Split the data into train, validation, and test set, and scaling the input features
7. Build 2-3 base models, including Random Forest, XGBoost, and Neural Networks. Use cross validation to check the models' performance
8. Select the XGBoost model and do hyperparameter tunning using AWS SageMaker tool
9. Select the best performed model as the final model, and check its RMSE and MAE score on the test set
10. Compare the model performance with the benchmark model
11. Analyze the feature importance on the final model

In this project, most of the time are spent on data cleaning and feature engineering. While the dataset is organized and relatively clean, how to handle missing value is still quite time-consuming. The missing value in different column could have different meaning, so it needs to be handled column by column. A challenging yet interesting task is to deal with the categorical feature which has more than hundreds of unique values. It is infeasible to direct transform such feature using one-hot encoding since it will bring in hundreds of new input features. The solution used in this project is to keep values that have relatively large proportion and mark all the smaller one as "Other", then apply one-hot encoding on the categorical features. In addition, the SageMaker greatly improves the work efficiency since it provides huge computing power and many built-in functionalities such as hyperparameter tuning.

Improvement

This project has room for improvements on feature selection. For example. The current method to select numerical feature is to calculate the correlation coefficient and investigate on features which

has small score (~0) via plotting and visually check if there is some no-linear relationship between the feature and label. This approach is feasible, but it could be possible that some important features are not selected or missing. Another method is to use all the existing features to build the model, check the feature importance of each input feature, and build a new model with features that have high importance scores.