Lantian Li

Project: Predicting Fashion Sales Quantity

Link: https://github.com/skerrysky/reach_fashion_sales_prediction

I completed a solo project that uses fashion product online transaction data to predict sales quantity. I used feature engineering and deployed the XGBoost model to make the prediction. There are two major challenges: 1) extract features from training data and relevant tables; 2) set appropriate XGBoost parameters for good model performance.

Case Scenario:

A French e-commerce company has gathered a large amount of data on its fashion products' online transactions from the last quarter. The fashion products include two types: leather goods and accessories. The company is trying to increase marketing investment and achieve a high ROI in the next quarter. To optimize the investment strategy, it wants to utilize the data to effectively predict sales quantity.

Data Description:

There are 4 data files. My first step is to read data and go over each csv file one by one:

1) Training data: train.csv (row 5298, col 15), including fixed product attributes: month after launch, product id, type (leather goods or accessories), gender (aimed at female, male or unisex), function, model name, material, color, French price, description in English, target sales (quantity of products sold)

2) Test data: test.csv (5166, 14), similar to training data, but will be used for tests.

3) Sales: sales.csv (145135, 97), a huge amount of sales records, including transaction attributes: transaction date, month, type, zone number, country number, product id, sales quantity, exchange rates (of different countries and times), social network posts, mentions, and impressions (at different times), brand positive, negative, and net sentiments (at different times).

4) Navigation data: navigation.csv (261399, 9), including online visit records: product id, visit day and month, website zone and country, web traffic source (e.g., email, natural search), product page views, number of products added to cart.

Lantian Li

Feature Engineering:

First, I extracted important features in two steps:

1. Extract features in training and test data sets:
    1. Calculate mean target sales quantity for products of different colors (group by other attributes), and merge this mean_target to training and test data
    2. Count the number of products of identical attributes (product function, model name, material, color), and merge these counts to training and test data
2. Extract features in navigation and sales data sets:
    1. Summarize page views for products of different traffic sources (e.g., email, natural search, etc. total of 6 different sources)
    2. Summarize sales quantity for products of different transaction types (total of 2 types)
    3. Summarize sales quantity for products in different zones (total of 5 zones)
    4. Summarize online performances for different products (page views, social network posts, mentions, and impressions, brand positive, negative, and net sentiments)

Second, I processed the data to prepare for modeling:

1. Slice the training data into 5 parts to prepare for cross validation
2. Merge all the summarized features into training data
3. Preprocess features by changing labels (e.g., “women” as “-1”) and log transformation of y (target sales quantity)
4. Repeat 2) and 3) for test data
5. Create a set for all features
6. Define a custom train-test split function for cross validation with XGBoost (select from a specific month)

XGBoost Parameters:

Next, I chose XGBoost (Extreme Gradient Boosting) for predictive modeling, because of its exceptional performances in predictive accuracy, robustness, regularization techniques, and efficiency. Set parameters

1. Objective is to do regression
2. Evaluation metric is RMSE
3. Boosting model is based on gbtree (gradient-boosted decision trees)
4. Set parameters for
    1. eta (0.025, slower learning and potentially better convergence)
    2. subsample (70% of the data will be used in each round)

3. colsample_bytree (70% of the features will be used in each tree)
4. num_parallel_tree (3 parallel trees to grow during training)
5. min_child_weight (25 as minimum sum of instance weight needed in a child to prevent overfitting)
6. gamma (5 as minimum loss reduction required to make a further partition on a leaf node to control tree complexity)
7. max_depth (3 as maximum depth of each decision tree in the ensemble to control tree complexity and overfitting)

Model Results:

Then I used configured parameters to generate model results for Month 1, Month 2, and Month 3. I set the maximum number of boosting rounds or iterations as 50000, and the early_stopping_rounds as 20. I created oof_m1 and oof_test_m1 to store predictions for validation and test data sets respectively for Month 1.

Eventually, I created test_m1 to store the final prediction for the test dataset by taking the mean of predictions from each of the five models.

I also repeated these steps for Month 2 and 3, and aggregated the predictions (oof_m1, oof_m2, and oof_m3) into the target and pred_target.

Afterwards, I calculated RMSE for different months and an overall RMSE for all months:
- month1: 0.4454 (deviate from the actual values by about 0.4454 units)
- month2: 0.5822
- month3: 0.7159
- overall: 0.5916

We need more business contexts to gauge how good the model performance is, which is the next step.

Last, I printed out the prediction results (predicted sales quantity for test data set) after inverse log transformation.

Challenges:

I overcame two major challenges in this project:

1) I need to understand the complex relations between different data tables and variables, so that I can decide on which features to extract for the modeling. More business contexts would also be helpful. For instance, I created a feature based on the product color since I suspected it could be

affecting sales quantity, but I might be wrong and overlooked some other important product attributes.

2) Tuning the parameters for XGBoost modeling is also challenging because it is a highly iterative process. I approached this by looking for the best values for each parameter step by step, and the search followed the order below (start with the most independent parameter eta). In each step, I tried different values with cross validation and picked the one with the best RMSE, and then moved to the next parameter to repeat the search.

1. Learning rate (eta)
2. Number of parallel trees (num_parallel_tree)
3. Tree-specific parameters:

- Maximum depth of tree (max_depth)
- Minimum child weight (min_child_weight)
- Subsample (subsample)
- Colsample_bytree (colsample_bytree)

4. Gamma (gamma)

I am proud of this project because feature engineering is a critical skill for machine learning, and XGBoost is a popular algorithm for a variety of machine learning tasks.