Application of Machine Learning Methods to Evaluate Collectibles

Category: Finance & Commerce

Team Members (SUNet ID): Xiaotian Lu (06605876)

Github Link: https://github.com/timothyxlu/CS229/

Abstract

The project explores SVM and tree models to apply to a multiclassification problem in the field of valuing collectibles. Data preprocessing techniques like normalization, one-hot encoding, and ordinal encoding are applied. A custom kernel function to handle mixed features and incorporate domain knowledge is developed. Bagging-based ensemble is examined to reduce variance.

1. Introduction:

Pricing strategy is an interesting area in commercial settings attracting attention from research community. Microeconomic theory provides us a general framework to analyze dynamic of supply and demand curves for common goods but fails to concretize a quantitative methodology to tackle collectibles. This project focus on a special type of collectibles that has limited supply for each customization in primary market. Producers usually set the same price for all similar products. However, in secondary market, consumers' valuation varies greatly from one customization to another, often impacted by artistic or cultural elements or consumer's behavior. Providing a model-based evaluation can help sellers to list their collectibles optimally in a secondary market when relevant transaction data is still scarce. This project develops and evaluates SVM and Decision Tree based pricing models for a popular product – "Funko Pop!".

2. Related Work

Prior research [1] exists in developing artwork pricing models for online art sales using text analytics. The author uses a dataset containing 150,000 artworks and reduces the price dimension to discrete intervals. Base features include categorical data like artist country, artwork medium and artwork style. kNN, SVM and Random Forest are used to develop multi-classification model. Then the author uses Paragraph Vector to convert product description to vectors, which is clustered using K-Means. The new feature is included in previous models to evaluate its effectiveness.

3. Dataset and Features

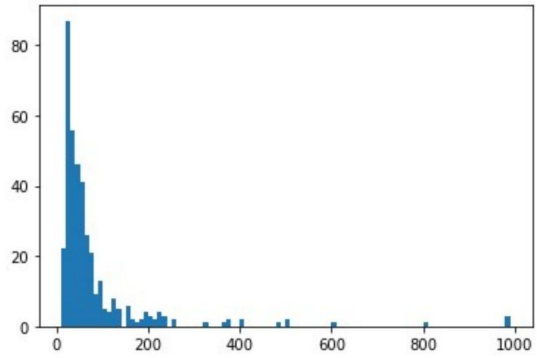
Stockx.com provides transparent collectibles transaction data in secondary market. A DevOps tool named Puppeter can be used to extract data from JavaScript heavy websites. Its python wrapper PyPpeteer is used in this project. "Funko Pop!" sells toys in different categories, such as "Disney" and "Marvel". For the purpose of this project, a subset of categories is included considering number of data points and whether the product is related to a fictional character. After excluding items with no transaction data, counts of items for each category are summarized below.

Category	Ad-Icons	Anime	Disney	Game-of-Thrones	Marvel
Count	43	198	37	11	69

Release date info is provided for only 108 items. Besides that, other product meta data available are product name and product picture. Product name provides some extra info about the item. For example, "Funko Pop! Animation DragonBall Z Beerus (Metallic) SDCC Figure #120" indicates the topic –

“DragonBall Z Beerus”, the material – “Metallic” and the special version – “San Diego Comic Con”. Category, topic, material, and special version are considered as base features. The topic info itself is suitable as direct input for modeling. I convert it to search volume to represent public interest in the topic. Google Trend provides only relative search volume. An alternative tool from “ahrefs.com/keyword-generator” is used to acquire absolute search volume for topic keywords.

The chart below shows the histogram of average sale prices. Prices greater than 1000 are clipped at 1000.

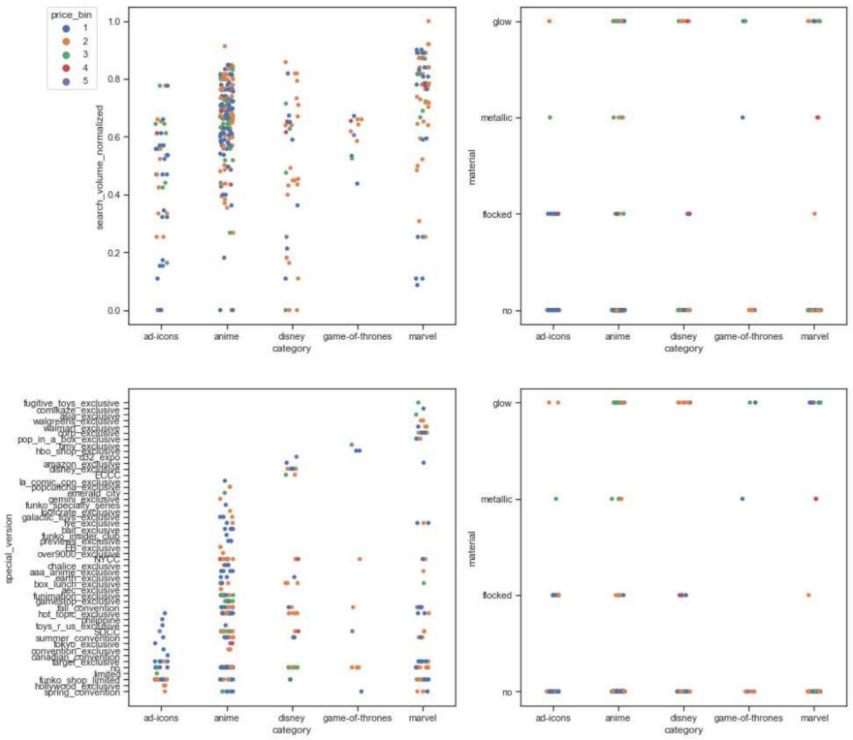


Picture 1. Histogram of Average Sale Price

As lower prices are clustered and higher prices are scattered, forecasting widely ranging continuous price with categorical input is not likely viable. On the other hand, forecasting price ranges is considered a reasonable goal in this commercial context. In this project, the number of bins is set to 5. As the price distribution is close to lognormal distribution, log prices are divided evenly into 5 bins.

Price Bin#	1	2	3	4	5
Range	[14,38.13]	[38.13,104.87]	[104.87,287.02]	[287.02,785.55]	[758.55,2150]
Item Count	146	153	46	10	4

Search volume values are approximately lognormally distributed. To make popular linear models perform better, this attribute is normalized by first taking log value and then using min-max scaler to scale value between 0 and 1. The plots visualize the relationship between multiclass labels and two of features.



4. Baseline Model and Metrics

In this step, a linear SVM model is applied to this multiclassification problem to establish the baseline. In the previous step, search volume variable is normalized. There are three other categorical variables. One-Hot Encoding is used to preprocess them so that linear SVM model can handle features properly.

In this step, we carefully consider the model selection strategy and metrics. In a typical cross validation configuration, the whole dataset is split randomly into train/test datasets. In our case, class frequencies are imbalanced where count of high price items is much smaller compared to low price items. Stratified K-Fold cross validation is used for imbalanced classification, where data of each class is split into train/test proportionally. With regards to metrics, weighted Precision/Recall/F1 scores are calculated to compare different models and hyperparameters. In addition, per class precision is also checked as we are interested to see if a model can learn from relatively small high-price training dataset. SVM does not support multiclassification natively. Two popular binarization strategies exist, One-vs-All and One-vs-One. Research [4] shows OVO is less sensitive to imbalanced data as it uses only relevant training data for each binary classifier. From the results below, it is noticeable that Linear SVM does not fit well with training dataset and its performance on high-price bins is very poor. With one-hot-encoding, it is not likely the data is separable in linear space of features.

Weighted Avg	Precision	Recall	F1
Train	0.625	0.608	0.576
Valid	0.408	0.482	0.440

Per Class Precision	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
Train	0.582	0.625	0.882	0	0.75
Valid	0.498	0.482	0	0	0

5. Extended Experiments

5.1 SVM with Kernel Function

A powerful tool to handle non-linearity in SVM is to use Kernel method. A popular choice is RBF kernel function which measures similarity in feature space. However, RBF is not suitable for categorical variables. We can show changing kernel from 'linear' to 'rbf' for the same SVM classifier does not improve performance. Marco [5] proposed a new kernel function for categorical variables. The main idea is data with more overlapping features have higher similarity score with feature value probability adjusted. With some modifications, we extend it to incorporate numerical variable and implemented a custom kernel function. The steps are summarized as following.

1. Use Ordinal Encoder to encode categorical features and calculate probability $P(Z_i)$
2. Calculate similarity score for search volume $U_1 = \exp\left(-\frac{(x_i - x_j)^2}{2\theta^2}\right)$
3. Calculate similarity score for categorical features (category) $U_2 = h\big(P_Z(z_i)\big)\ if\ z_i = z_j\ else\ 0$
4. $h(z) = (1 - z^{\alpha})^{1/\alpha}$ is a convex function for $0 < \alpha < 1$
5. Apply step 3 to other features (material, special_version) and get U_3 and U_4
6. Use external weights to calculate composite kernel value $k = \sum_{l=1}^{4} w_l U_l$

With external weights as hyperparameter, we can impose domain knowledge of features. We know search interest and special version are more important for collectible. The weights for this experiment are set to be [2, 1, 1, 3]. From the results below, we can see with the new kernel, SVM is better at capturing high-price patterns in training data. Performance on test data is still a problem.

Weighted Avg	Precision	Recall	F1
Train	0.662	0.630	0.602
Valid	0.390	0.460	0.421

Per Class Precision	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
Train	0.611	0.638	0.882	0.625	1
Valid	0.485	0.453	0	0	0

5.2 Vanilla Decision Tree

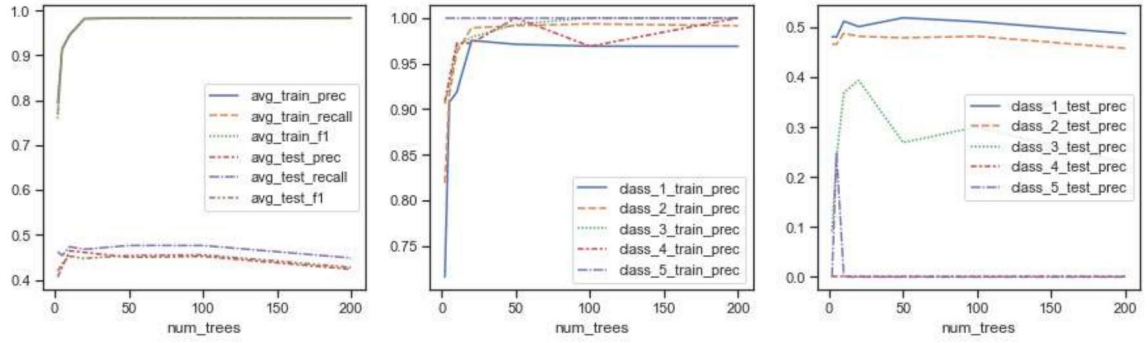
Now we change to a popular non-parametric model called Decision Tree. Unsurprisingly, Decision Tree achieves very high scores on training dataset and similar scores on valid dataset. It indicates overfitting, which is a typical problem for Decision Tree. Also, it is worth noting Decision Tree can predict high-price items in test data to some extent. This probably indicates Decision Tree models are better at handling categorical features.

Weighted Avg	Precision	Recall	F1
Train	0.984	0.983	0.983
Valid	0.408	0.415	0.409

Per Class Precision	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5
Train	0.951	1	1	1	1
Valid	0.477	0.452	0.137	0	0.333

5.3 Bagged Decision Tree

To resolve overfitting issue in vanilla decision tree model, Bagged Decision Tree is a Bagging-based ensemble model that trains a number of uncorrelated trees from random sampled input and predict using majority vote. In this experiment, we investigate its performance with varying num_trees hyperparameter. From the results below, the average scores are peaked when num_trees is around 100. We can see scores on test dataset are improved compared to vanilla decision tree. A few non-standard bagging algorithms are suggested in literature [7] to handle imbalanced classification problem. However, scikit learn does not allow for overriding bagging algorithm to test the effectiveness of advanced bagging techniques.



6. Conclusion and Future Work

In this project, we are faced with two challenging problems: class imbalance and mixed features. SVM with custom kernel that incorporates domain knowledge is a better parametric model. Tree model is in general better at handling categorical features. And bagging-based ensemble can reduce variance with

little impact on bias. In the end, I look at data to understand why high-price items are not predicted well by the models I have experimented so far. I found two potential factors. Search volume is subject to how topic is represented. Topic with more words has lower search volume. For example, ‘’Toy Story Buzz Lightyear” and “Buzz Lightyear” refer to the same thing but the latter has much higher search volume. Another hurdle of predicting high-price items is lack of useful features. Manual investigation of discussions in fan-based online forum like funkopop forum in reddit.com indicates some words in online post like “extremely” and “rare” in related posts are linked to high price items. However, at the time of this project, extracting such data is proved to be more difficult than expected. If data of high quality and larger size are available, the problem can be revisited in the future and natural language models can be applied.

7. Contributions

The team consists of only one member. The work is generally divided into three steps. First, exploring available data as well as viability of data extraction method. Second, setting up infrastructure to acquire data and clean data. Third, applying different machine learning methods and evaluate their performance.

Reference:

[1] Powell, L., Gelich, A., & Ras, Z. W. (2019). Developing artwork pricing models for online art sales using text analytics. In T. Mihálydeák, F. Min, G. Wang, M. BanerjeeIvo, I. Düntsch, Z. Suraj, & D. Ciucci (Eds.), International Joint Conference on Rough Sets (pp. 480-494). Springer. https://doi.org/10.1007/978-3-030-22815-6_37

[2] Lee, Lin & Wahba. Multicategory Support Vector Machines. http://www.stat.wisc.edu/~wahba/ftp1/lee.lin.wahba.04.pdf

[3] Jason B. How to Fix k-Fold Cross-Validation for Imbalanced Classification. https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/

[4] Michal Z. (2020). Performance Analysis of Binarization Strategies for Multi-class Imbalanced Data Classification.

[5] Marco A. (2013). An investigation into new kernels for categorical variables.

[6] Mikel G. et al. (2011). A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches.