Dorothy Modrall Sperling
Machine Learning 1
22/01/2021

PREDICTING PRICE CATEGORIES OF SKINCARE PRODUCTS IN SCIKIT-LEARN

I. INTRODUCTION

The skincare industry is extremely profitable - the global market for cosmetic skincare in 2020 was estimated at 145.3 billion USD.[1] However, how skincare companies price their products seems to be a closely-guarded secret. For example, a 16-ounce tub of *La Mer* cream is priced at 2,475 USD on the *La Mer* official website, while the much-cheaper *Nivea Creme* moisturizer has a nearly identical ingredient list.[2] *The Ordinary* is another brand that has become hugely popular for its affordable skincare products and no-frills ingredients. *The Ordinary* is able to keep costs down by minimizing marketing spending compared to other skincare companies, and only using ingredients that have been shown to be effective (e.g., retinoid and vitamin C).[3] This begs the question – when you buy a skincare product, are you paying for brand and marketing costs, or for the ingredients and formulations?

Of course, when most people pay a lot for skincare, they expect high-quality ingredients and formulations that yield the best results. It is in the interest of luxury skincare companies to perpetuate the theory that a higher price tag is synonymous with higher-quality ingredients and performance. In this paper, I will use various machine learning algorithms in Scikit-Learn to predict the price of a product. The first part of my paper is exploratory, comparing binary classifiers trained on data with and without brand information to predict if a product is “cheap” or “expensive”. The second part of my paper attempts to build multiclass classifiers to sort products into 1 of 4 price categories – “cheapest”, “cheap, “expensive”, and “most expensive”.

My hypothesis is that including the product ingredient list in the training data will not cause the classifiers to perform better, i.e., that a product’s ingredients do not contribute meaningfully to its price.

---

[1] “Global Cosmetic Skin Care Industry (2020 to 2027) - Market Trajectory & Analytics.” GlobeNewswire News Room, "GlobeNewswire", 25 Sept. 2020, www.globenewswire.com/news-release/2020/09/25/2099275/0/en/Global-Cosmetic-Skin-Care-Industry-2020-to-2027-Market-Trajectory-Analytics.html.

[2] “Moisturizer for Dry Skin: La Mer Official Site.” *Crème De La Mer*, www.cremedelamer.com/product/5834/12343/moisturizers/creme-de-la-mer/moisturizer-for-dry-skin#/sku/26717.

[3] Bond, 22 Feb 2019 “How Does The Ordinary Stay So Affordable?” Adore Beauty, 8 Oct. 2020, www.adorebeauty.com.au/the-ordinary/guide/how-does-the-ordinary-stay-so-affordable.

Instead, I expect that "brand" will be a much more informative feature for predicting the price of a product.

The dataset I use in this project is the "Skincare Products and their Ingredients" dataset, which contains information about 1138 products extracted from the online retailer *LookFantastic.com.*[4] The dataset includes the product names, the product types (moisturizer, body wash, serum, etc.), uncleaned lists of the products' ingredients, and the product prices in GBP.

My project consists of 5 parts:

1) Binary classification using ingredient list, amount of product (i.e., contents), and product type.
2) Binary classification using brand, ingredient list, amount of product, and product type.
3) Binary classification using brand and product type (without any information about the product amount or ingredients).
4) Multiclass classification using brand, ingredient list, amount of product, and product type.
5) Multiclass classification using brand and product type (without any information about the product amount or ingredients).

II. METHODS

1. PREPROCESSING

To preprocess the data and prepare it for an algorithm, I first converted the "price" column into a column of floating-point numbers, removing the "£" symbol. I also added a "brand" column and filled in the brand of each product manually. Then, I added a "contents" column, containing the amount of product for in item, by using a regex to extract the milliliter, gram, or kilogram amount from "product_name". For example, the item "Acorelle Pure Harvest Body Perfume - 100ml" has the value "100" in the "contents" column (the limitations of this process will be discussed in my "Limitations" section.) Unfortunately, the "contents" column contained 155 missing values, where the regex was not able to find a product amount in "product_name". I manually searched for and filled volume amounts into the "contents" column to replace these missing values. I decided to create this "contents" column

[4] Ward, Erin. "Skincare Products and Their Ingredients." *Kaggle*, 31 Oct. 2020, www.kaggle.com/eward96/skincare-products-and-their-ingredients.

because if the ingredient list had any predictive value, it is logical that the *amount* of those ingredients would have an effect on the product price as well. I kept the columns "brand", "contents", "product_type", and "ingredients" as my predictive features in my dataset.

I used a column transformer to prepare the dataset. "product_type" had 14 unique values that were encoded as strings, so I used OneHotEncoder() on this column to get the corresponding dummy variables. I used StandardScaler() on the column with "contents", containing numeric data, to center the column values on mean 0 with a standard deviation of 1 (StandardScaler() was not used in experiments with decision trees, since decision trees do not benefit from features being scaled).

Finally, I used CountVectorizer() on the "ingredients" column. I chose CountVectorizer()for the "ingredients" column because one-hot-encoding each ingredient would cause errors if unseen ingredients appeared in the test set's "ingredients" column. CountVectorizer() simply ignores vocabulary it has not been trained on. For the count vectorizer, I defined a custom tokenizer to remove unnecessary characters or artefacts of the dataset, splitting on commas (the ingredients were separated by commas, e.g., "water, butylene glycol, …"). These transformers were compiled using the function make_column_transformer(). The column transformer was combined in a pipeline with each machine learning algorithm.

2. BINARY CLASSIFICATION PROCEDURE

I built binary classifiers to put skincare products into one of two categories – "cheap" (under 18.90 £) and "expensive" (over 18.90 £). Admittedly, this distinction is arbitrary. I obtained it by inspecting the inter-quartile range of the price column, and finding that 50% of products were under 18.90 £, and that 50% were over 18.90 £. The goal of this section was simply to see if any information about price could be obtained from a combination of the predictors "brand", "contents", "product_type", and "ingredients".

In the first part of my experiment, I included all predictors except "brand". That is, I used the predictors "contents", "product_type", and "ingredients", excluding "brand". In the second part of my experiment, I included all predictors: "brand", "contents", "product type", and "ingredients". In the third part of my experiment, I removed all information about both the ingredients and the amount of product. Instead, I only used the "brand" and "product_type" column. I suspect that "content" does not provide useful information about price. For example, the average serum (coming in 30-50ml bottles) can

cost over 30 euros, while body wash can come in 1-liter bottles and only cost a fraction of the former's price.

In binary classification, I tested the accuracy of the perceptron, logistic regression, decision tree, K-nearest neighbors (KNN), and support vector machine (SVM) classifiers. I also included three ensemble learning models, such as a voting classifier, bootstrap aggregation (bagging), and random forests classifiers in predicting a product's price category. Hyperparameter tuning was performed by randomized grid search on the logistic regression, decision tree, KNN, SVM, and random forests classifiers (GridSearchCV was extremely slow and evidently too computationally expensive).

Randomized grid search was not used on the perceptron, voting, and bagging classifiers. I chose to give the ensemble voting classifier the 3 best-performing algorithms (all fitted with the optimal hyperparameters obtained during the randomized grid search on these algorithms). The bagging classifier used decision trees that were fitted with the optimal hyperparameters obtained during randomized grid search on the decision tree classifier.

3. MULTICLASS CLASSIFICATION PROCEDURE

I built multiclass classifiers to put skincare products into one of four categories – "cheapest" (<9.95 £), "cheap" (9.95£-18.90£), "expensive" (18.90£-31.25£) and "most expensive" (>31.25 £). I obtained these by inspecting the inter-quartile range of the price column, and using each quartile as a category. The goal of this section was to see if a specific price range could be predicted from a combination of the predictors "brand", "contents", "product_type", and "ingredients".

By the time I had finished the binary classification experiments, it had become clear that "brand" was a useful predictor of price category. Therefore, in this section, I performed multiclass classification using all predictors: "brand", "contents", "product_type", and "ingredients". I compared these results with multiclass classification involving "brand" and "product_type", with no information about the ingredients or amount of product included.

In multiclass classification, I tested the accuracy of the logistic regression, decision tree, K-nearest neighbors (KNN), and support vector machine (SVM) classifiers. I also included the same three ensemble learning models: a voting classifier, bootstrap aggregation (bagging), and random forests.

I did not include the perceptron because making the perceptron accommodate multiclass labels was cumbersome and likely would not have yielded good results. Instead, I explored different multiclass strategies in logistic regression and SVM. The decision tree, K-nearest neighbors, ensemble voting, bootstrap aggregation (which used decision trees) and random forests classifiers did not have to be explicitly modified to accommodate multiclass labels.

I used one-vs-rest and multinomial classification in my logistic regression models. In one-vs-rest, the logistic regression algorithm calculated the probability of each point being of a certain class compared to the other classes.[5] In multinomial logistic regression, the log odds of the labels are predicted instead.[6]

The one-vs-rest strategy is identical in logistic regression and SVMs, but since SVMs do not naturally output probabilities, I also used a one-vs-one strategy in the SVM classifier. In the one-vs-one strategy, the SVM classifier splits the multiclass classification problem into one binary classification problem per pair of classes (e.g. “cheap” vs “most expensive”, “expensive” vs “cheapest”, etc.).

Hyperparameter tuning was performed by randomized grid search on the logistic regression, decision tree, KNN, and random forests algorithms. The bagging classifier was fitted with decision trees with the optimized hyperparameters found with RandomizedSearchCV(), and the voting classifier was fitted with the three best-performing algorithms with optimized hyperparameters. I did not perform RandomizedSearchCV() on the SVM classifiers because the OneVsOneClassifier() and OneVsRestClassifier() did not have easily-optimizable hyperparameters.

III. RESULTS AND DISCUSSION

1. BINARY CLASSIFICATION RESULTS

Figure 1. For each classifier, the 10-fold cross-validation score was calculated. These 10-fold cross-validation scores are reported below. The perceptron, ensemble voting, and bagging classifiers did not undergo hyperparameter tuning, though the bagging classifier’s *n_estimators=* hyperparameter was optimized. The best accuracies of each experiment iteration are highlighted in yellow for visibility.

[5] Raschka. pp.31.

[6] “Multinomial Logistic Regression.” UCLA Institute for Digital Research & Education Statistical Consulting Group. UCLA, stats.idre.ucla.edu/stata/dae/multinomiallogistic-regression/.

Binary Classification Cross-Validation Accuracy:				
	All predictors excluding "brand"	All predictors including "brand"	Only "brand" and "product_type"	**Average Accuracy**
Perceptron	0.72832481 +/- 0.02821925	0.75758738 +/- 0.05468463	0.79569480 +/- 0.07587360	**0.760536**
Logistic Regression*	0.72971014 +/- 0.05775174	0.77082268 +/- 0.04156579	0.83254476 +/- 0.03750473	**0.777693**
Decision Tree*	0.71359761 +/- 0.05653384	0.71359761 +/- 0.05653384	0.75034101 +/- 0.04844292	**0.725845**
K-Nearest Neighbors*	0.66807332 +/- 0.04508327	0.63864024 +/- 0.06699692	0.83107417 +/- 0.04440287	**0.712596**
SVM*	0.73706309 +/- 0.05433613	0.76496164 +/- 0.05684514	0.81489770 +/- 0.05037799	**0.772307**
Ensemble Voting	0.74737852 +/- 0.06284060	0.77960358 +/- 0.05874637	0.82664109 +/- 0.05271464	**0.784541**
Bootstrap Aggregation	0.70624467 +/- 0.06196272	0.71210571 +/- 0.05847654	0.79136829 +/- 0.05362545	**0.736573**
Random Forests*	0.77231458 +/- 0.05088843	0.78554987 +/- 0.05758467	0.77966752 +/- 0.04238528	**0.779177**
Average Accuracy	**0.7253383**	**0.740358589**	**0.802779**	

* = optimized with RandomizedSearchCV

Yellow highlight = best performance

Red highlight = worst performance

On average, classifiers performed the best with only "brand" and "product_type" as predictors, with an average classifier prediction of roughly 80.3%. When the "ingredients" column was included, the random forests classifier performed the best, whether or not "brand" was present. I suspect that the random forests classifier performed the best when ingredients were present because vectorizing the ingredient lists produced extremely long feature vectors. In random forests, only *d* features can be sampled at each node of the tree without replacement, where *d* 's default value = sqrt (# of features).[7] This might help reduce the number of dimensions and reduce overfitting when working with extremely long feature vectors.

The extremely long feature vectors might also explain the poor performance of KNN when the "ingredients" column was included. The feature space was likely so sparse that neighboring data points

[7]Raschka. pp.245.

were too far away from each other to make meaningful predictions (i.e., the dataset was suffering from the curse of dimensionality).[8]

The best-performing model, on average, was the ensemble voting classifier, with an average of ~78.5% accuracy. This makes sense, since the ensemble voting classifier was trained with the three highest-performing non-ensemble machine learning algorithms (logistic regression, SVM, and perceptron), and made predictions by majority vote. However, the best accuracy was obtained by logistic regression when using only "brand" and "product_type", with an accuracy of 83.3%. Overall, parametric models including SVM, logistic regression, and perceptrons performed better than non-parametric models including KNN and decision tree classifiers.

2. MULTICLASS CLASSIFICATION RESULTS

Figure 2. For each classifier, the 10-fold cross-validation score was calculated after performing a randomized grid search to optimize classifier's hyperparameters. These 10-fold cross-validation scores are reported below. The SVM, ensemble voting, and bagging classifiers did not undergo hyperparameter tuning, though the bagging classifier's *n_estimators=* hyperparameter was optimized. The best accuracies of each experiment iteration are highlighted in yellow for visibility.

Multiclass Classification Cross-Validation Accuracy:			
	All predictors with brand included	Only brand and product type included	**Average Accuracy**
Logistic Regression	**OVR:** 0.53448423 +/- 0.05401498 **Multinomial:** 0.54479966 +/- 0.05566250	**OVR:** 0.64618500 +/- 0.05258931 **Multinomial:** 0.64471441 +/- 0.05192325	**OVR: 0.590335 Multinomial: 0.594757035**
Decision Tree*	0.47271952 +/- 0.05647606	0.53154305 +/- 0.04647858	**0.502131285**
K-Nearest Neighbors*	0.42719523 +/- 0.04240450	0.56973572 +/- 0.06485797	**0.498465475**
SVM	**OvR:** 0.52133419 +/- 0.03726042 **OvO:** 0.53604007 +/- 0.03811698	**OvR:** 0.59916880 +/- 0.04694893 **OvO:** 0.64320119 +/- 0.06592216	**OvR: 0.560251495 OvO: 0.589621**
Ensemble Voting	0.54341432 +/- 0.04054984	0.64910486 +/- 0.06812677	**0.59626**
Bootstrap Aggregation	0.54763427 +/- 0.04893143	0.52853794 +/- 0.05664643	**0.538086**
Random Forests*	0.56084825 +/- 0.04741750	0.52860188 +/- 0.06064640	**0.544725065**
Average Accuracy	**0.520941082**	**0.5934214**	

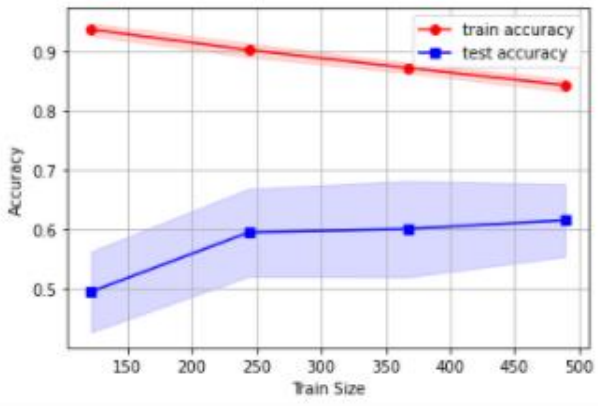
[8] Raschka. pp.107.

* = optimized with RandomizedSearchCV

Yellow highlight = best performance

Red highlight = worst performance

On average, classifiers performed better with only “brand” and “product_type” as predictors, with an average classifier prediction of roughly 59.3%. However, these results are dismal compared to the modest accuracy scores obtained in the previous section.



A quick visualization of the learning curve for one of the learning algorithms (SVC in this case) might shed some light onto the problem. It is clear that the training accuracy is quite high with small standard deviations, while the test accuracy is low. Because there is a large gap between training and test accuracy, it is clear the at this model suffers from high variance, meaning that the complexity of the model needs to be reduced or that more data needs to be collected.[9]

The best-performing model, on average, was the ensemble voting classifier, with an average of ~59.6% accuracy. This was expected since the voting classifier was trained with the three highest-performing non-ensemble machine learning algorithms (in this case, the logistic regression with one-vs.-rest, SVM with one-vs.-one, and a decision tree classifier), and made predictions by majority vote. KNN, again, was the worst-performing, probably because even without the “ingredients” column, the feature space was too sparse to make meaningful predictions about the specific price range of products.

IV. LIMITATIONS

There are many problems with the experimental design of this project. One of the most important problems is the encoding of ingredients and their names with CountVectorizer(). There was variation in what the same ingredients were called from product to product. For example, in one product, an ingredient was called “fragrance”, and in another, “parfum”. When CountVectorizer() is applied, these identical ingredients will be separated into two different feature columns, making the matrix more sparse. It is possible that there are hundreds of duplicate ingredients in separate columns because of

[9] Raschka. pp.202.

different naming conventions or even spelling – I am not familiar enough with chemistry to catch these redundancies. In future experiments, ingredient names should be thoroughly researched to look for alternative names for the same product that appear in ingredient lists of other products in the dataset. The ingredient lists should then be standardized to have one ingredient name for one ingredient and vice versa.

A second problem with this experiment was that the training and test datasets were not stratified by brand. There were many brands represented by only one product in the dataset. Because of the CountVectorizer() preprocessing step, if one of these products ended up in the test dataset, the model wouldn't be able to use its brand to make a prediction about the price category. Collecting more data would solve the problem of brands that are represented by only one product in the dataset. Gathering data on other products by these brands would help models make more predictions about a brand's typical price range.

A third problem with this experiment was that the order of ingredients within the ingredient list matters. For example, if "water" appears at the beginning of the ingredient list, it means that there is more water in that product than any other ingredient.[10] Ingredients at the end of the ingredient list have the lowest concentration in that product. CountVectorizer() does not preserve information about the order of ingredients, so information about the relative amounts of each ingredient is lost. Potential insight into effect of ingredients on price is therefore obscured. In the future, a CountVectorizer() should be designed to split a list of ingredients, record the index of the ingredient in the list, and fill feature columns with respective indices to indicate the place of the ingredient in the list. Or, an algorithm that preserves information about order, like an LSTM, could be used to preserve information about ingredient orderings.

A fourth problem with this experiment is the "content" column. When I created this column, I stripped product amounts (like 100ml, 50g, etc.) of their units ('g' and 'ml') and treated them as the same unit to get a measurement. Grams are a measurement of mass, and milliliters are a measurement of volume. 1 gram is equivalent to 1 milliliter of water, so in this experiment, I assumed that all products had roughly the same viscosity as water when I removed the "ml" and "g" labels. Of course, this is an inaccurate assumption, but I did not have access to the viscosity measurements of each product to make

[10] "Understanding Cosmetics Ingredients Labels: Paula's Choice." Paula's Choice, www.paulaschoice.com/expert-advice/skincare-advice/skin-care-how-tos/understanding-cosmetics-ingredients-labels.html.

a conversion of grams to milliliters. It was likely a mistake to make this column, and in future experiments, it should not be included.

A fifth problem with this experiment is that this dataset contains no information about the different amounts of money these brands spend on marketing in general. The amount of money spent on marketing is recouped in the price of the product, so marketing budget influences the price of a product. I was tempted to see "brand" as a proxy for the amount of money spent on marketing, but even within brands, there is likely to be variation in marketing budget by product line. For example, one product line might be more aggressively marketed than another within a brand – *L'Oreal*, for instance, has the *L'Oreal Luxe* line, which is priced higher on average than the rest of the company's products.[11] Therefore, "brand" is not a straightforward proxy for marketing budget and therefore not an ideal predictor of price category. A solution for this problem is to gather more data points per brand and within different product lines.

V. CONCLUSION

The results of the experiments with binary and multiclass classifiers show that brand, indeed, is somewhat predictive of price. Predicting the price with the ingredient list of a product yielded worse accuracies, whether or not brand was included as a predictor. The best accuracies were achieved in binary classifiers that only used the features "product_type" and "brand". Multiclass classifiers that attempted to separate products into four price ranges with these same features, however, performed very poorly. Because models were unable to perform a more complex task like multiclass classification, it is clear that additional data is needed to make these models more robust. Although the results of this project were disappointing, at least I now know that I should not assume that a hefty price tag means higher-quality skincare ingredients.

[11] L'Oréal. "L'Oréal Group : L'Oréal Luxe Division." *L'Oréal*, L'Oréal, 1 Jan. 1AD, www.loreal.com/en/loreal-luxe/.

VI. CITATIONS:

Bond. 22 Feb 2019 “How Does The Ordinary Stay So Affordable?” *Adore Beauty*, 8 Oct. 2020, www.adorebeauty.com.au/the-ordinary/guide/how-does-the-ordinary-stay-so-affordable.

Gio. “Is Nivea Creme Really A Dupe For Creme De La Mer?” *Beautiful With Brains*, 26 Aug. 2020, www.beautifulwithbrains.com/dupes-creme-de-la-mer-nivea-creme/.

“Global Cosmetic Skin Care Industry (2020 to 2027) - Market Trajectory & Analytics.” *GlobeNewswire News Room*, "GlobeNewswire", 25 Sept. 2020, www.globenewswire.com/news-release/2020/09/25/2099275/0/en/Global-Cosmetic-Skin-Care-Industry-2020-to-2027-Market-Trajectory-Analytics.html.

L'Oréal. “L'Oréal Group : L'Oréal Luxe Division.” *L'Oréal*, L'Oréal, 1 Jan. 1AD, www.loreal.com/en/loreal-luxe/.

“Moisturizer for Dry Skin: La Mer Official Site.” *Crème De La Mer*, www.cremedelamer.com/product/5834/12343/moisturizers/creme-de-la-mer/moisturizer-for-dry-skin#/sku/26717.

“Multinomial Logistic Regression.” *UCLA Institute for Digital Research & Education Statistical Consulting Group.* , UCLA, stats.idre.ucla.edu/stata/dae/multinomiallogistic-regression/.

Raschka, Sebastian. *Python Machine Learning*. Packt Publishing Limited, 2015.

“Understanding Cosmetics Ingredients Labels: Paula's Choice.” *Paula's Choice*, www.paulaschoice.com/expert-advice/skincare-advice/skin-care-how-tos/understanding-cosmetics-ingredients-labels.html.

Ward, Erin. “Skincare Products and Their Ingredients.” *Kaggle*, 31 Oct. 2020, www.kaggle.com/eward96/skincare-products-and-their-ingredients.