Classifying winners of Allure award

By Anastasia Gorina

Introduction and process

Each year experts in **Allure magazine** give their red seal of approval to several dozens of products. Usually it indicates high quality and effectiveness, so people know that the buy is worth their money.

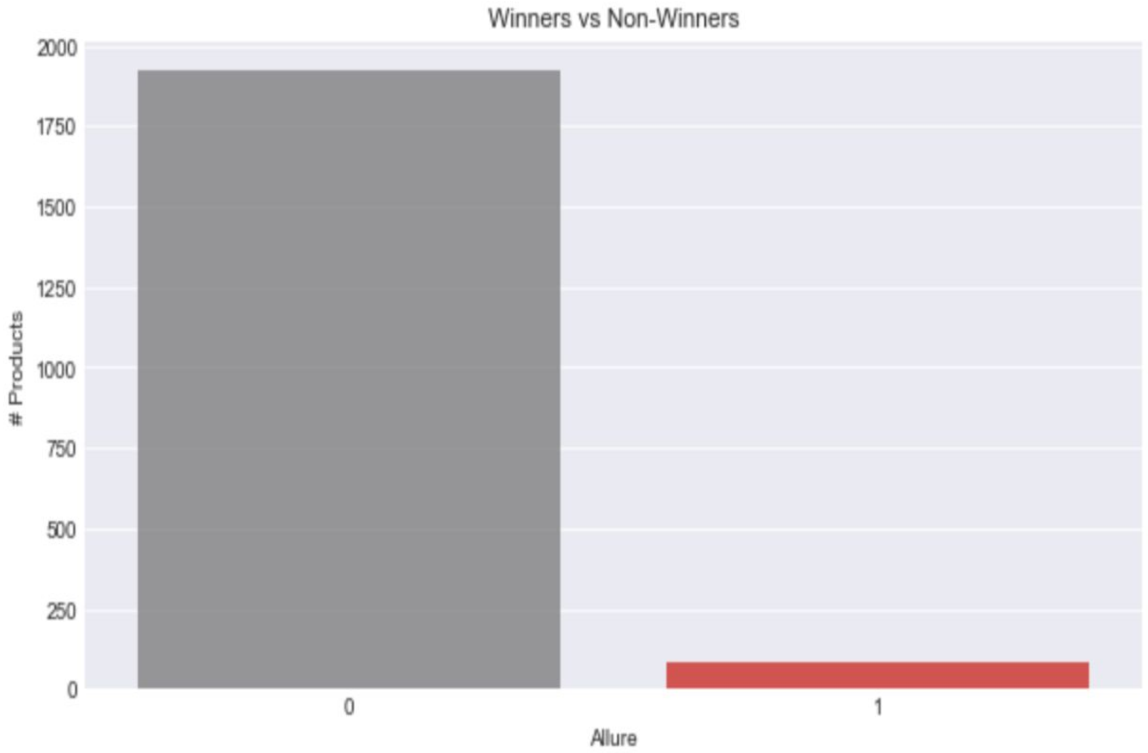
I used machine learning algorithms to spot the winners by using the information about the product from Sephora.com: **price**, **number of reviews**, **number of likes**, and ‘**clean beauty**’ seal.

→  → plotly → scikit learn

The Problem

Since only **81** product out of 2000 sold at Sephora won the Allure Beauty Award-2019, there was a severe **class imbalance**.

Naturally, my first model (logistic regression) predicted that every given observation is a not a winner with **96% accuracy** but **no recall.**



Handling Class Imbalance

Traditional approach: random undersampling/oversampling with sklearn built-in methods, SMOTE, Tomek links. All of the above failed to significantly improve my models.

My approach:

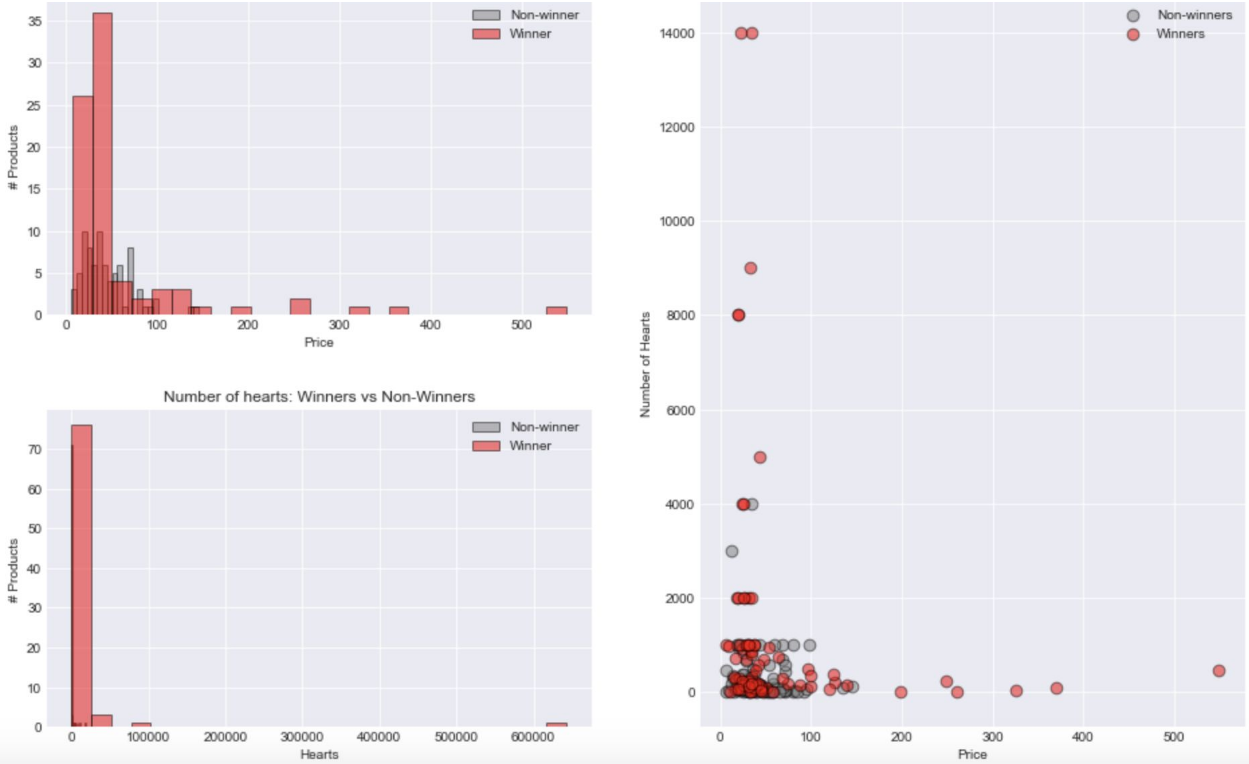
- Divide data into two parts, winners and non-winners.
- Shuffle the rows of non-winners dataframe.
- Crop the non-winners dataframe to match the dimensions of the winners dataframe
- Concatenate the two and shuffle again

Equally represented classes

Analysis of the new data with equally represented classes demonstrated some significant difference between the winners and non-winners, especially in the number of 'hearts' and reviews. Therefore, it is possible for the model to spot the winner.

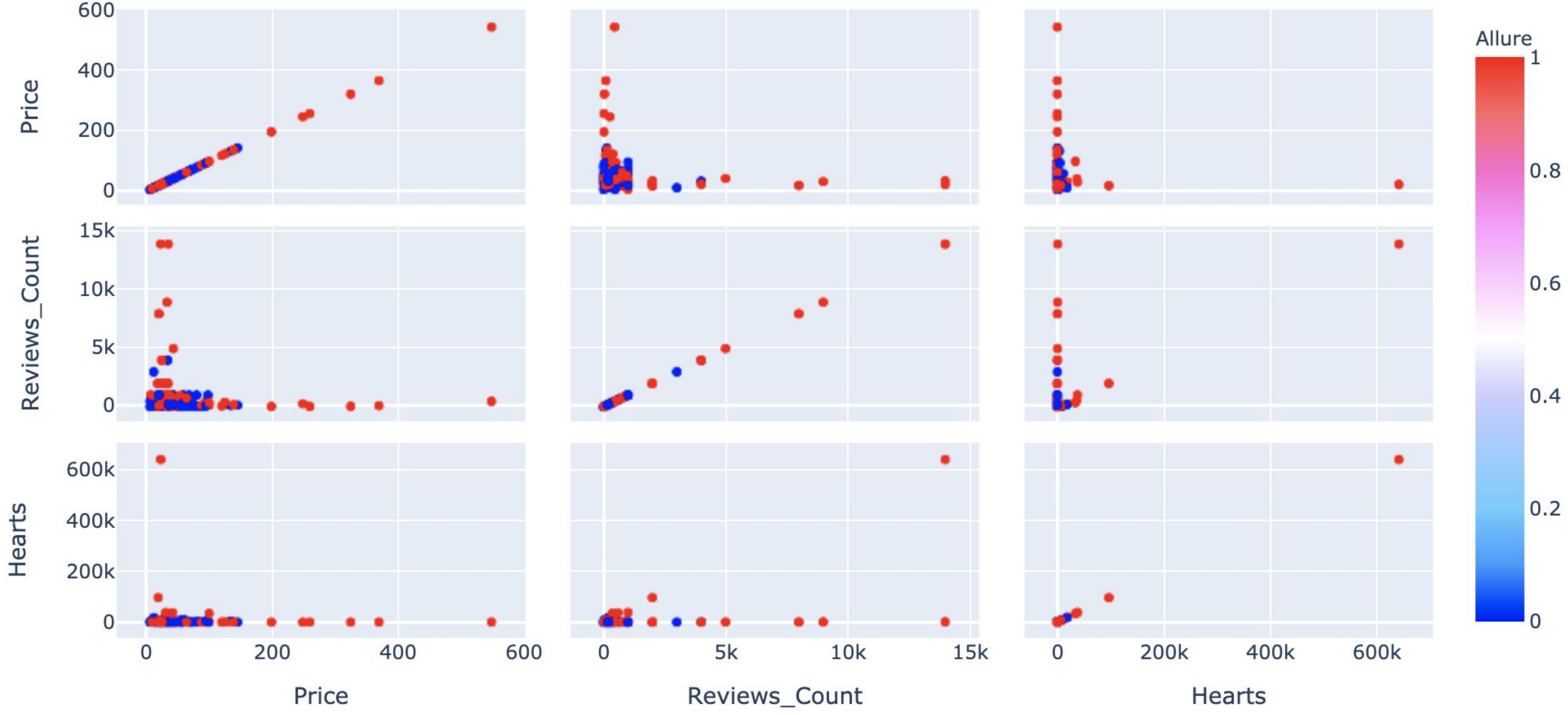
	Price		Reviews_Count		Hearts		Clean	
	mean	std	mean	std	mean	std	mean	std
Allure								
0	44.877778	27.042864	351.604938	604.625353	739.154321	2648.526085	0.123457	0.331010
1	62.059136	85.578456	1260.061728	2669.376066	10866.644444	72255.936525	0.135802	0.344713

Visualizations



Allure award-winning products have higher variance and a lot of outliers in each category

Scatter Matrix



Final Model

After handling the class imbalance problem with double shuffling and undersampling, all of the models (Logistic regression, KNN, Naive Bayes, XGBoost) improved in recall and accuracy. However, the best model turned out to be **XGBoost**.

Training Accuracy Score	89.38%
Validation Accuracy Score	93.88%
Precision, non-winners	0.89
Precision, winners	1.00
Recall, non-winners	1.00
Recall, winners	0.88
F1	0.94

$$\begin{pmatrix} 24 & 0 \\ 3 & 22 \end{pmatrix}$$

Training Accuracy Score	92.25%
Validation Accuracy Score	96.97%
Precision, non-winners	1.00
Precision, winners	0.95
Recall, non-winners	0.93
Recall, winners	1.00
F1	0.97

$$\begin{pmatrix} 14 & 1 \\ 0 & 18 \end{pmatrix}$$

Conclusion

- In order to build the best performing model I needed to select the correct way to handle the class imbalance, although even after that **not all models** made good predictions (e.g. **Naive Bayes** was only **2%** better than random picking).
-
- For better and broader analysis I would also look at the products that are **not sold at Sephora**. Some mass-market brands can also win, so the dataset can be extended.
-
- Also, in the future I can use **NLP** to analyze the user reviews as well as the expert opinions to make even more accurate prediction and compare **winning** products to those that were **nominated**, but didn't win.