Classifying Customer Location in StockX Transactions

Kevin Le

Brown University, Data Science Initiative

Github: https://github.com/kevin-j-le/StockXConsumerRegionClassification

December 2, 2020

Introduction

With the advent of online platforms for authentication, the global and US sneaker resale industries have become multi-billion dollar industries almost overnight. StockX, one of the world's largest sneaker resale platforms, is a key player in the global resell market. Every day, StockX processes millions of transactions for resold sneakers and stores massive quantities of sneaker sales data as a result.

The dataset used in this project comes from StockX's 2019 Data Contest and consists of a fixed percentage of Yeezy and Off-White x Nike sales in the US from everyday between September 2017 and February 2019. In total, the data contains 99,956 data points and 8 features. Features include *'Order_Date', 'Brand', 'Sneaker_Name', 'Sale_Price (USD)', 'Retail_Price (USD)', 'Release_Date', 'Shoe_Size'*, and *'Buyer_Region'*. The goal of this project is to classify a consumer's *'Buyer_Region'* based on the values of the other features. Because *'Buyer_Region'* contained 51 unique labels, I consolidated the buyer region into four different regions in the US (West, Midwest, South, Northeast) and created a new column *'US_Region'* to reduce the number of labels. I then performed the same machine learning models with respect to *'US_Region'* and *'Buyer Region'* to compare performance. Because our target variables are categorical, this problem type is classification. Classifying the regions where transactions are processed can be applicable in business contexts by possibly uncovering insights behind consumer behavior. For example, sellers can determine regions where consumers purchase sneakers at the highest resale prices and target those markets. Previous projects have predicted sneaker resale price, while others have discerned the best/worst time to resell a sneaker.

Exploratory Data Analysis

Before beginning EDA, I cleaned the data and explored basic properties of the data. After calling the describe() method to obtain descriptive statistics for continuous variables, I discovered that the mean sale price was $446.63, while the mean retail price was $208. I then retrieved the counts for unique values in categorical values using pandas' value_counts function, where we learned that our target variable *'Buyer_Region'* had 51 unique values. To reduce the number of labels, I consolidated each region into a larger US region (West, Midwest, South, or Northeast) in accordance with the US Census Bureau and created a new column, *'US_Region'*. I reduced the number of classes in hopes of improving model interpretability.

Figure 1 explores the market share of each sneaker. Of all sneakers, the adidas Yeezy Boost 350 V2 Butter had the highest market share, with over 10% of transactions being made for that sneaker. Figure 2 visualizes the proportion of sneaker sales by US Region. Our dataset is balanced with respect to our target variable, with the West making the most sales with little over 33% of all sales and the Midwest making the least with 12.9%.. Figure 3 explores the distribution of sneaker resale prices per US Region using a violin plot. Interestingly, the resale prices for each region are nearly identically distributed. The probability densities appear small because each region contains sales that are massive outliers from the rest of the data. Figure 4 contains a stacked bar plot visualizing brand sales by US region. Not only are brand sales proportions almost equal among all regions, they closely mirror overall brand sales proportions, shown in Figure 5.

Methods

The dataset is identically and independently distributed without any group structure, as observed in Figure 3. Although order/release dates are highly descriptive of the data, the dataset is not time-series, since every transaction is a distinct event that is not influenced by other data points. All categorical features, which included 'Brand', 'Sneaker_Name', 'Release_Date', were preprocessed using a OneHotEncoder. 'Shoe_Size' and 'Order_Date' were scaled using a MinMaxScaler, since they had clear minimum and maximum values. 'Retail_Price' and 'Sale_Price' were scaled with a StandardScaler. The procedures described below were applied separately with respect to both of our target variables, *US_Region* and *Buyer_Region*.

The machine learning pipeline implemented in this project was meant to optimize for faster training times, while also accounting for variability inherent in splitting and non-deterministic ML methods. Each model was trained using a stratified random sample of our feature matrix. This sampling technique needed to be stratified to ensure that our sample is representative of our original dataset. To be exact, the models trained on 25,000 randomly sampled rows of our dataset. I sampled rows in this manner with hopes of reducing training times, while maintaining the same rigor as training on the full dataset.

These randomly sampled rows were passed into the pipeline, which was defined as its own function. The pipeline accepted a feature matrix, target variable, preprocessor (ColumnTransformer), machine learning algorithm of choice, parameter grid, and random state, while returning the best model parameters and validation score. In essence, the pipeline acts as a cross-validator that selects the best parameters given a subset of the data. The first step of the pipeline splits the data into 5 folds using a StratifiedKFold, to ensure every fold is similarly distributed. A pipeline object containing the ColumnTransformer object and ML algorithm is created using make_pipeline( ). This pipeline object is then passed into a GridSearchCV object, along with our parameter grid and StratifiedKFold. Accuracy is the chosen evaluation metric because the data is balanced. The model with the best parameters was reapplied to the rest of the data. In this procedure, the data was split 80/20 into a train and test set using train_test_split. The algorithm was then refitted on the training set and delivered test scores. This procedure was repeated for 5 random states.

Four ML algorithms were applied to the dataset: logistic regression, linear support vector machine, random forest classifier, and k-nearest neighbors classifier. For logistic regression, I tuned *penalty* type (l1 vs. l2) and the regularization coefficient *C*, which was logspaced from 10^{-3} to 10^{4}. Instead of regular support vector classifiers, I opted for linear support vector classifiers because they train more quickly and are more suitable for large datasets. Unlike regular SVC, linear SVC also does not support a parameter for *gamma*. Hence, only the regularization coefficient *C* was tuned in a log-spaced interval from 10^{-3} to 10^{4}. For our random forest classifier, *max_depth* was tuned through linearly spaced values from 10 to 110, and *max_features* were tuned over a range of 0.5, 0.75, and 1.0. For K-neighbors classification, *n_neighbors* parameter was tuned over a range of 500, 1000, and 2000 nearest neighbors. I applied these algorithms using the procedure described in the previous paragraph and tried to predict both of our target variables, *Buyer_Region* and *US_Region* separately. Lastly, I collected the average test accuracy scores and standard deviations for each ML algorithm to measure their performance and spread.

Results

Figures 6 and 7 describe the performance of each algorithm with respect to a baseline accuracy score when predicting *US_Region* and *Buyer_Region* respectively. Baseline scores were calculated as if the model predicted the most populous class for every guess. For example, the baseline score for *US_Region* is 0.334 because the most populous class (West) makes up 33.4% of the data. When predicting *US_Region*, all models performed just marginally better than the baseline accuracy by 10 standard deviations. Many of the models showed standard deviations as low as 0.003 and test accuracies ranging from 0.349 to 0.37. With an accuracy of 0.37, random forest classifiers best predicted *US_Region*, followed by K-nearest Neighbors with a score of 0.353, linear SVC with 0.35, and logistic regression with 0.349.

Relative to the baseline, the models performed worse when predicting *Buyer_Region* versus predicting *US_Region*. With a baseline score of 0.194, logistic regression had an accuracy of 0.196 +/- 0.003. Linear SVC performed with an accuracy of 0.197 +/- 0.002. K-nearest neighbors suffered the worst performance with a score of 0.194 +/- 0.002, and random forests again performed best with an accuracy of 0.216 +/- 0.004.

Global feature importances were then calculated using perturbation feature importances and global SHAP values, whereas local feature importances were determined by creating force plots from SHAP explainers. They were interpreted with respect to the random forest classifier, which performed the best compared to the other algorithms. Figures 8 and 9 describe global feature importances using perturbation feature importances and global SHAP values, respectively. These metrics both identify the same three most important features but in different order. In descending order of importance, the perturbation feature metric listed *Shoe_Size* as most important, then

Order_Date, and *Sale_Price*. On the other hand, the global SHAP values determined *Order_Date* to be most important, then *Sale_Price* and *Shoe_Size*. In the context of random forests, these three features contributed the most to the splits determined by decision trees. Meanwhile, the least important features consisted of various release dates and sneaker models. Surprisingly, two release dates and two sneaker models appeared in our list of most important features, some of which were more important than *Brand*. Local feature importances largely mirror the global feature importances, with the most important models positively contributing to predictions.

Outlook

Given the relatively poor performance of our ML models on the classification problem, I propose some factors and weak spots that could possibly be behind the performance. Although I initially thought that the features did not have enough predictive power, three features play a significant role in predicting customer location. For future purposes, I could apply the ML algorithms on a reduced feature matrix containing only *Order_Date*, *Sale_Price*, *Shoe_Size*, and *Retail_Price*. Including all features in our matrix could have introduced noise into our models. Another reason behind the models' poor performance could be in the nature of the data itself. When referring to our figures from EDA, we notice almost identical distributions for sale prices and brands. I suppose it would be difficult for the models to ascertain differences between rows that map to different labels.

Some crucial flaws also lie in the splitting and sampling strategy. Only 25% of the feature matrix was passed into the function for cross-validation. To add, the rows of the target variable were not sampled in a stratified manner, unlike the feature matrix rows. In the future, I could train the models on larger subsamples of the data and find a way to sample from the target variable in a stratified manner. The methods applied were meant to accommodate fast training times and robust performance, but perhaps I should use more robust methods, even at the expense of time. In the future, I'd like to apply XGBoost. Lastly, the dataset only refers to sales for Yeezy and Off-White sneakers; I think it would be useful to use transactional data for shoes of other brands as well, as it could provide insight on the hype of certain sneakers in certain places.

Figures

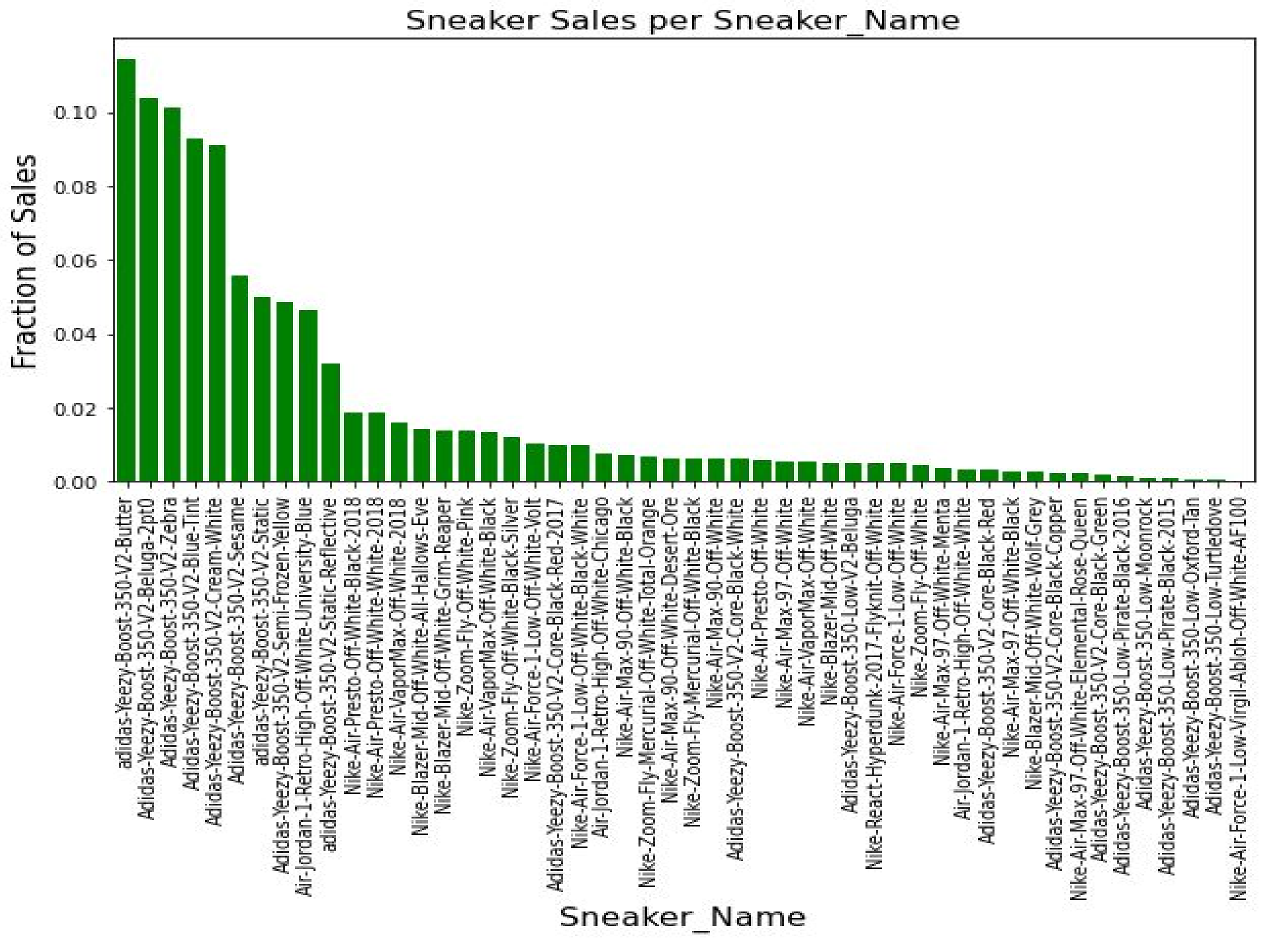

Figure 1. Fraction of sales per sneaker model. The top 7 selling sneakers are Yeezy, while the highest selling Off-White sneaker is the Off-White x Nike Air Jordan 1.

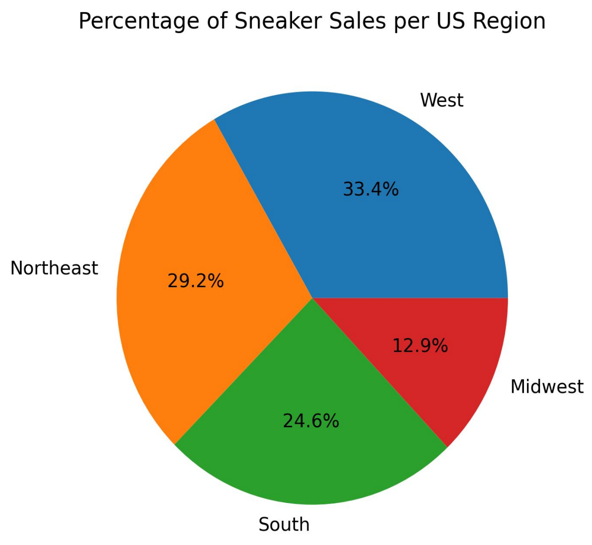

Figure 2. Percentage of sneaker sales per US Region. The data reflects a balanced distribution of sales per US region.

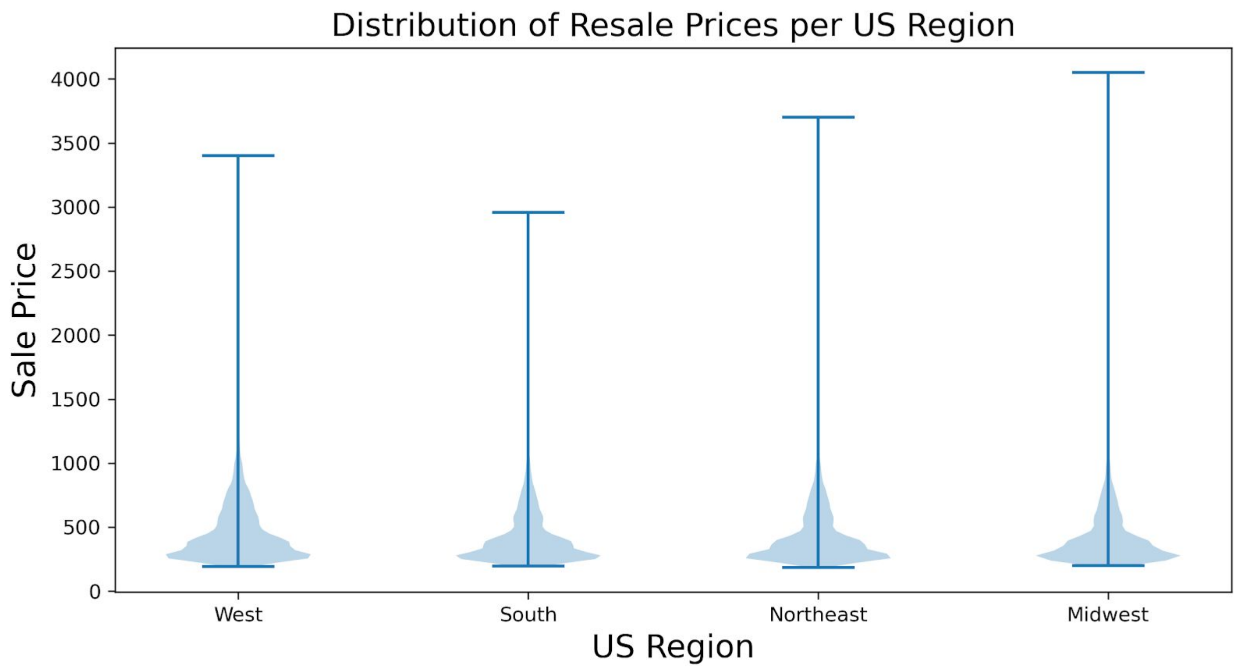

Figure 3. Distribution of resale prices per US region. Resale prices are almost identically distributed across regions and includes outliers.

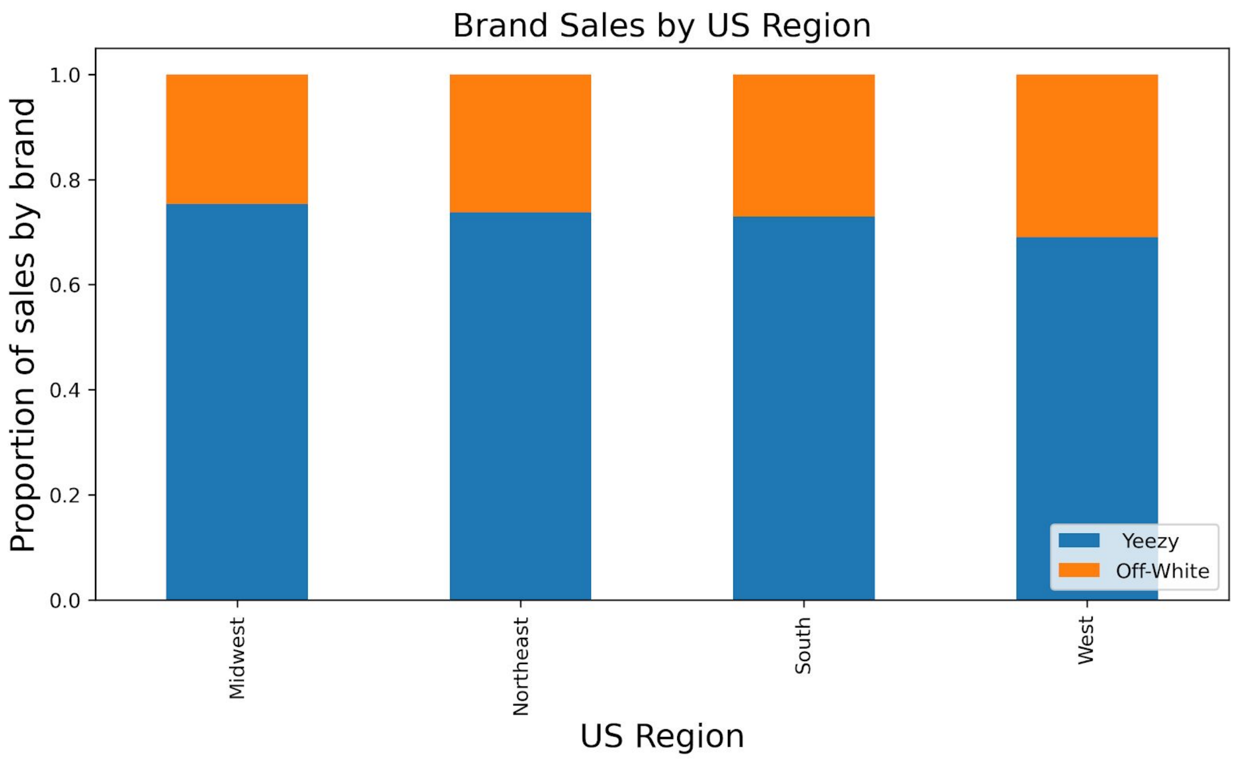

Figure 4. Proportion of sales per brand with respect to the US region. This closely reflects the overall proportion of sneaker sales between Off-WhitexNike and Yeezy.

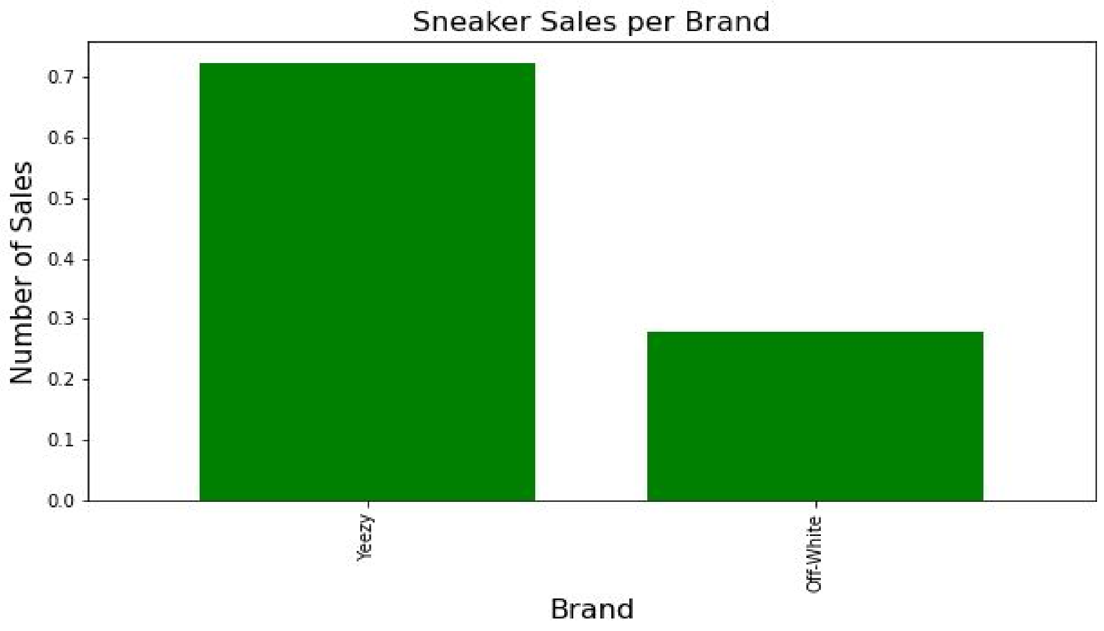

Figure 5. Normalized sneaker sales per brand. Yeezy sales more than doubled Off-White x Nike sales.

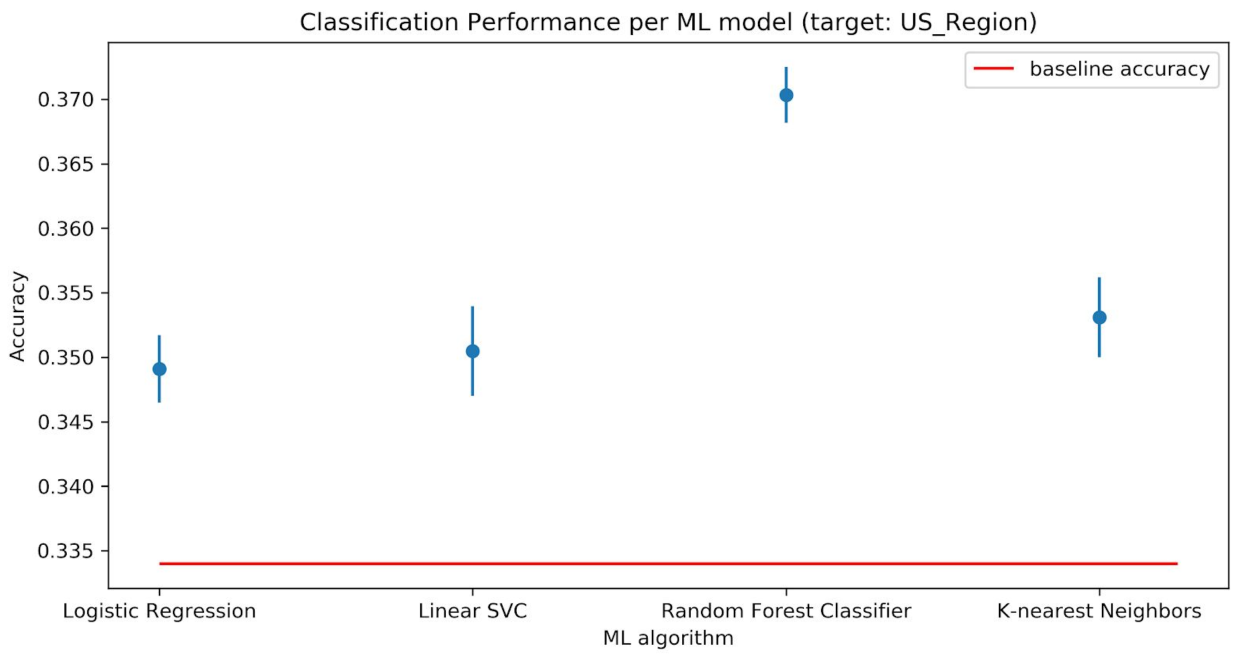

Figure 6. Accuracy scores of ML models when predicting the US region. Baseline score: 0.334.

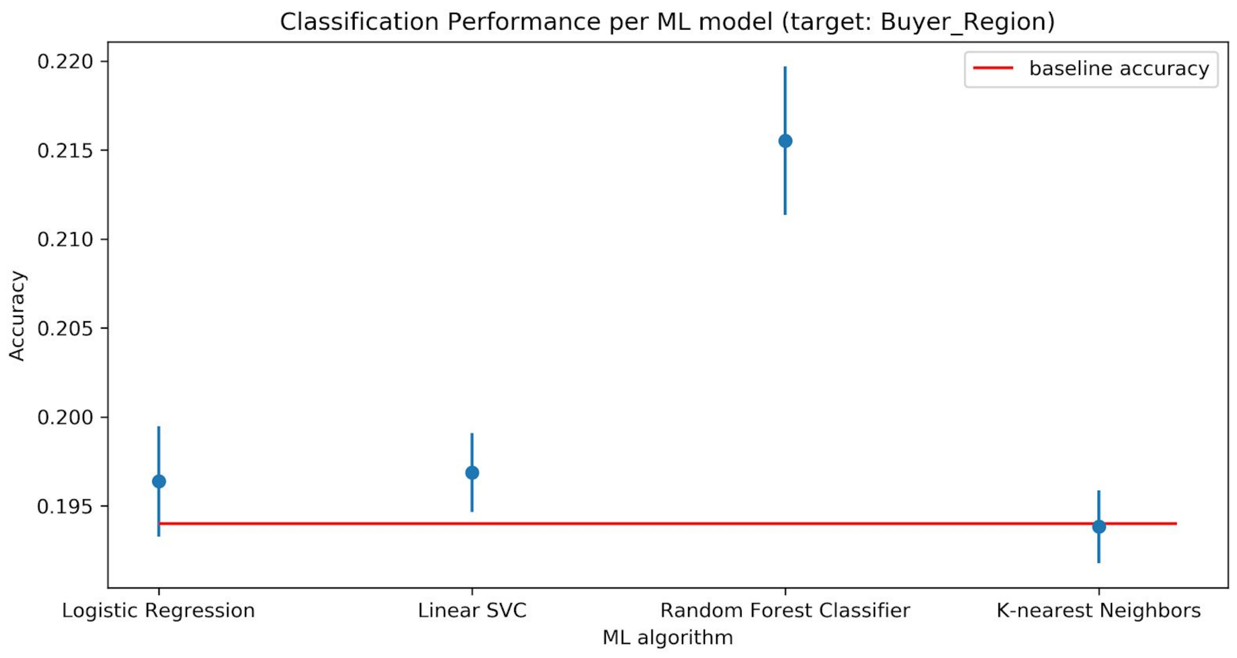

Figure 7. Accuracy scores of ML models when predicting the Buyer Region. Baseline score equal to 0.194.

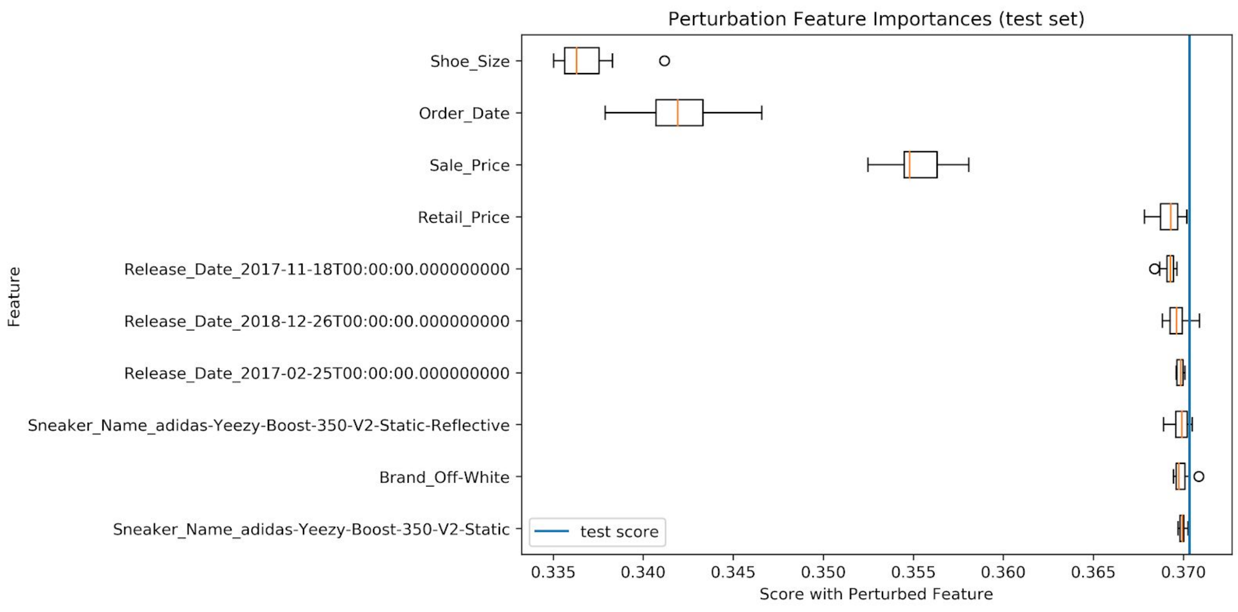

Figure 8. Top 10 most important features using the permuted features method.

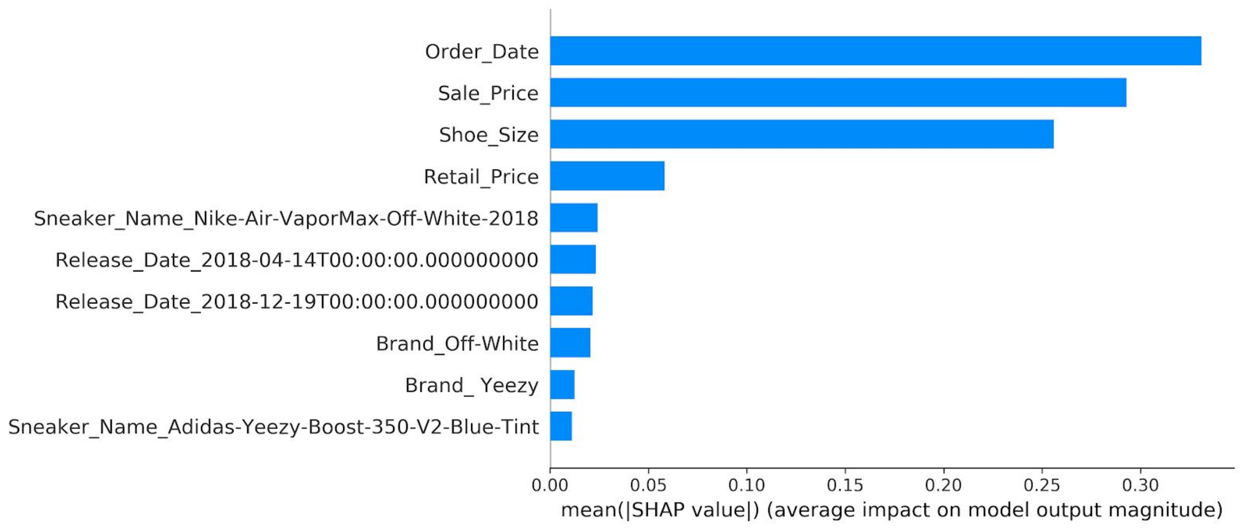

Figure 9. Top 10 most important features using SHAP values.

References

Einhorn, Jesse. “Attention Data Nerds: The StockX Data Contest Is Back.” *StockX News*, StockX, 7 Mar. 2019, stockx.com/news/the-2019-data-contest/.

“List of Regions of the United States.” *Wikipedia*, Wikimedia Foundation, 5 Oct. 2020, en.wikipedia.org/wiki/List_of_regions_of_the_United_States.

“The Global Sneaker Resale Market Could Reach $30 Billion by 2030.” *Yahoo! Finance*, Yahoo!, finance.yahoo.com/news/global-sneaker-resale-market-could-reach-30-billion-by-2030-cowen-191003371.html.

Norman, Logan. “StockX Sneaker Price Regression.” *Kaggle*, Kaggle, 6 Sept. 2020, www.kaggle.com/logannorman/stockx-sneaker-price-regression.

“Sk-Learn Documentation.” *Scikit*, scikit-learn.org/stable/.