Chocolate Bar Ratings Prediction

Xiang Fu, Kayla Wu

Introduction

Believe it or not, chocolate is ranked the most popular candy in the world (Bradford, 2022). In fact, the average American consumes 3 chocolate bars per week. For our final project, we were interested in exploring expert ratings of over 1,700 individual chocolate bars from a Chocolate Bar Ratings dataset via Kaggle (Tatman, 2017). In other words, we will be determining which parameter has the greatest impact on chocolate bar ratings using the following parameters/columns: (1) company manufactured, (2) broad bean origin, (3) percentage of cocoa, and (4) specific bean origin, to train our machine learning model. Additionally, we imported a country to continent dataset via Kaggle to generate more meaningful insights in tandem with our existing dataset (Gokhale, 2018). The primary objective of this project was to determine which parameter has the greatest impact on chocolate bar ratings.

Methodology

First and foremost, data cleaning was an essential step for finalizing our parameters of choice. Firstly, we renamed all 9 column names for clarity, brevity, and format. Next, we addressed the issue of missing, redundant, inconsistent, and abbreviated values by standardizing the formats through pattern-matching as well as replacing and dropping values. Our dataset is comprised of 1,795 total observations. To accomplish this, we did feature engineering to transform inputs to more useful formats:

1. **Cocoa Percent**: Converted string to float values. Given that Python wouldn't be able to identify the special character "%" as a percentage value, our approach was to replace it with a float number.
2. **Company**: Stripped the parentheses. Created and merged a new column named "New_Company," which contains the entries of "Company" that contain more than 20 observations. Otherwise, they were named "other."
3. **Bean Type**: The missing values from the "Bean Type" column were replaced with "Not specified." Based on our observations, the largest "Bean Type" group was encoded as '\xa0' by default.
4. **Broad Origin and Specific Origin**: Replaced abbreviated countries and states with full country names.
    a. Downloaded and imported 'countryContinent' (Gokhale, 2018) dataset from Kaggle to extract the country's corresponding continent and merged this subset to our entire dataframe.

Afterward, we incrementally built our Decision Tree and Random Forest regression model to train and improve our model's ability to accurately predict chocolate bar ratings.

1. **Training and Testing Data**: First, we use random.seed() to confirm that the same random numbers were generated each time we run the code. This is to ensure that the results were reproducible. Then, we set up the testSize variable and used len(df) to split the data into 80% training data and 20% testing data. Then a variable called testIndices was set up with random.sample() method to randomly select 20% of the data and set it aside as the test data. We then used .sort() method to sort the test data in ascending order. Lastly, df.iloc[testIndices] and df.drop(testIndices) sent the data to dfTest and dfTrain, respectively. The resulting test data set contained 359 observations and 5 attributes, while the training data contained 1,436 observations and 5 attributes.
2. **Get_Dummies**: We used the function "get_dummies" to convert categorical variables into dummies, or indicator variables.
3. **Decision Tree:** We experimented with setting the following parameters of the DecisionTreeRegressor constructor with a maximum depth of 5. We discovered this function by Matthew Mayo and KDnuggets, which expresses the decision tree model as a function, which helped us visualize the steps of our tree model. Later, we also used the traditional way, tree module from sciki-learn, and matplotlib.pyplot to plot both the original decision tree model (Exhibit 1) and the optimized decision tree model (Exhibit 2).
4. **Random Forest:** We imported the RandomForestRegressor class from the sklearn.ensemble module. This class implements a random forest regressor, which is a type of ensemble learning model that can be used for regression tasks. Then, we create d an instance of the RandomForestRegressor class and stored it in the rfm variable. The n_estimators parameter specifies the number of decision trees in the forest and the random_state parameter sets the random seed for the model. The model was trained on the training data by using the fit method of the RandomForestRegressor class. This method takes two arguments: the input data (X_train) and the target data (y_train), which contain the features and labels for the training data, respectively. For plotting the tree in random forests, we imported the tree class and the matplotlib.pyplot module, which was used to plot the decision tree. Then, we created a new figure with a specified size using the plt.figure method. Lastly, the original random forest model (Exhibit 3) and optimized random forest model (Exhibit 4) were plotted by using the plot_tree method of the tree class. This method takes several arguments: the decision tree to be plotted (rfm.estimators_[0]), the names of the features in the data (X_train.columns), the names of the classes in the data (y_train), and a Boolean value indicating whether the tree should be filled in (filled = True).
5. **Parameter Tuning**: To find the best parameter values for both the original decision tree model and the original random forest model, we used GridSearchCV from scikit-learn and entered the model and its parameters. Essentially, it is a cross-validation method. The predictions are made after extracting the best parameter values.

   For the decision tree, we set up the parameter_grid with the variables 'max _depth', 'max_features', 'min_samples_leaf', and 'min_samples_split'. After setting up the dt_grid

and GridSearchCV, the program calculates the optimal values for parameters, which are 'max_ depth': 8, 'max features': 10, 'min_samples_leaf': 8, and 'min samples split': 4.

For random forest, we set up the parameter_grid with the variables 'max_depth', 'max features', 'min_ samples_leaf', 'min samples split', 'n_estimators'. After setting up the rf_grid and GridSearchCV, the program calculated the optimal values for parameters, which are 'max_depth': 15, 'max_features': 6, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 100. These parameter values were passed to the optimized random forest model, optimized_rfm.

6. **Rating Predictions:** To provide data for the accuracy calculation, we added a 'Rating' column to a test set of data, and then used a decision tree model (optimized_dtm) to predict the ratings for the test set. The predicted ratings were then compared to the actual ratings in a dataframe, and the results were displayed. The code then repeated this process using a random forest model (optimized_rfm) instead of the decision tree model.
7. **Feature Importance:** To determine the top 3 variables that had the greatest impact on chocolate bar ratings, we created a function called top3_variables. The function took a model and a training dataset as input and outputted the top 3 variables based on their feature importance values. This function calculated the feature importance values using the feature_importances_ attribute of the model. The indices of the top 3 variables were then determined using the argsort method of numpy, and the names of the variables were printed along with their importance values. Then we called the top3_variables function for four different models: the original decision tree model (dtm), the optimized decision tree model (optimized_dtm), the original random forest model (rfm), and the optimized random forest model (optimized_rfm). For each model, the function printed the top 3 variables that had the greatest impact on chocolate bar ratings prediction.

Results

In order to evaluate the performance of both the original and optimized versions of the decision tree and random forest model, we created a function, with 3 inputs, model, test_features, and test_labels, respectively. We used two statistical measurements to measure the performance of the model: RMSE to show how far predictions fall from measured true values using Euclidean distance, and R2 Score to measure the amount of variance in the predictions explained by the dataset. For accuracy, we found the errors by calculating the absolute value of prediction values and subtracting test_labels. Then, we calculated the mean absolute percentage error (MAPE), which shows the average magnitude of error produced by a model, or how far off predictions are on average. These values are calculated by taking the absolute value of the difference between the actual and predicted values, dividing it by the actual value, and then multiplying it by 100. The mean of all of these values is then taken to get the MAPE. The accuracy is then calculated

by subtracting the MAPE from 100. All in all, our Decision Tree model generated an 86.7% accuracy rate and the Random Forest model generated an 87.0% accuracy rate (Exhibit 5).

After the feature importance process, we found that the “cocoa percentage” parameter had the greatest impact on chocolate bar ratings, with a 33-37% feature importance.

Conclusion

In summary, we both learned that when data cleaning, the workflow should be as follows: (1) *inspect* unexpected, incorrect, inconsistent data early in the pipeline (2) *clean* the discovered anomalies, (3) *verify* correctness before reporting changes made In other words, we discovered that it's important to inspect the data for any unexpected, incorrect, or inconsistent values early in the process, as this can help to identify potential issues that may affect the accuracy of the model. Once any anomalies have been discovered, the next step is to clean the data by either correcting or removing these values. This may involve a range of different techniques, such as interpolation, extrapolation, or outlier detection and removal. It's important to carefully consider the best approach for dealing with each individual anomaly, as choosing the wrong method could introduce new errors or biases into the data. After the data has been cleaned, it's important to verify the correctness of the changes that have been made. This may involve comparing the cleaned data to the original data or using other techniques such as cross-validation to ensure that the changes have not introduced any new errors or biases. It's also important to document the changes that have been made, as this will help to ensure that the data cleaning process can be easily reproducible in the future. Overall, following this workflow will help ensure that the data is as clean and accurate as possible, which is essential for training a high-quality machine learning model.

Exhibits

Exhibit 1: Original DTM

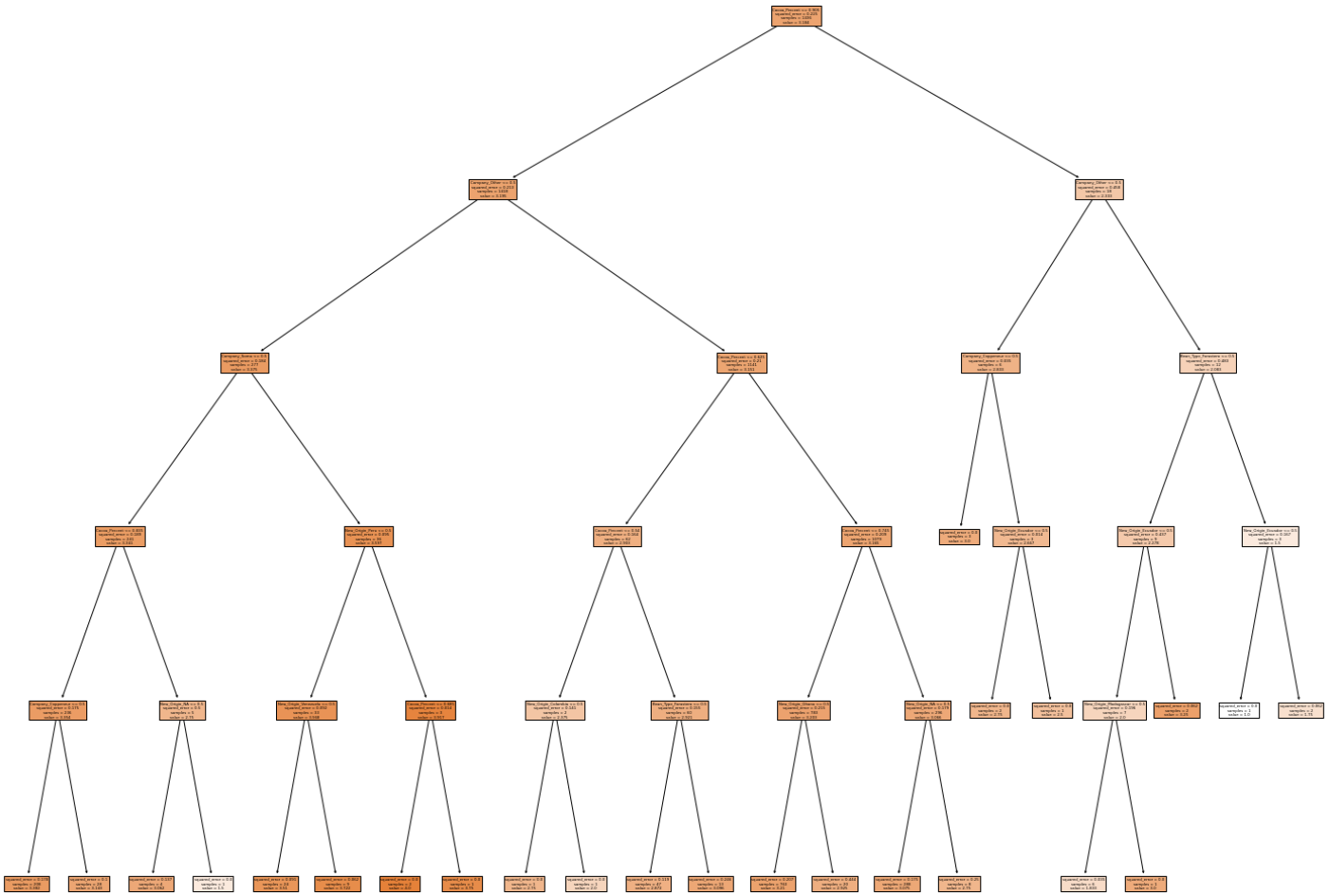

Exhibit 2: Optimized DTM

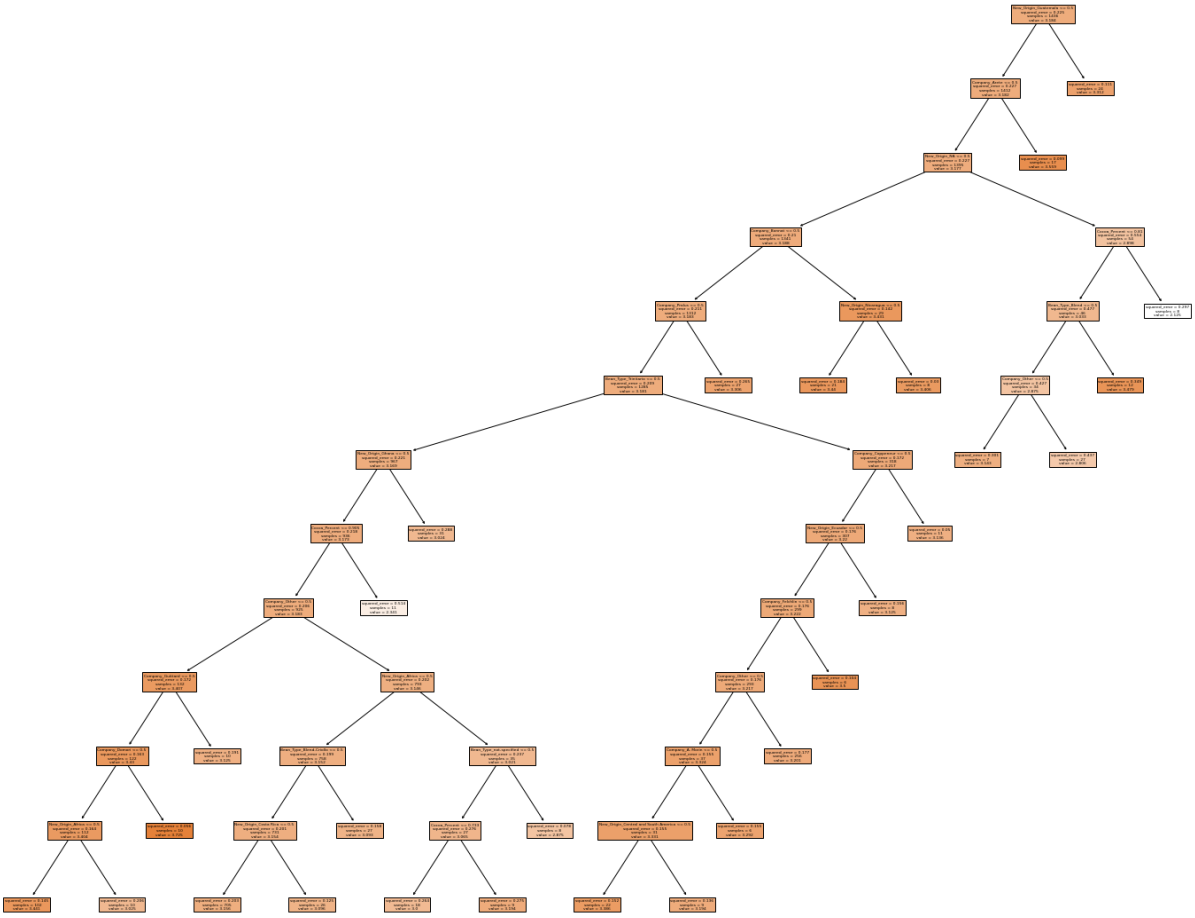

Exhibit 3: Original RFM

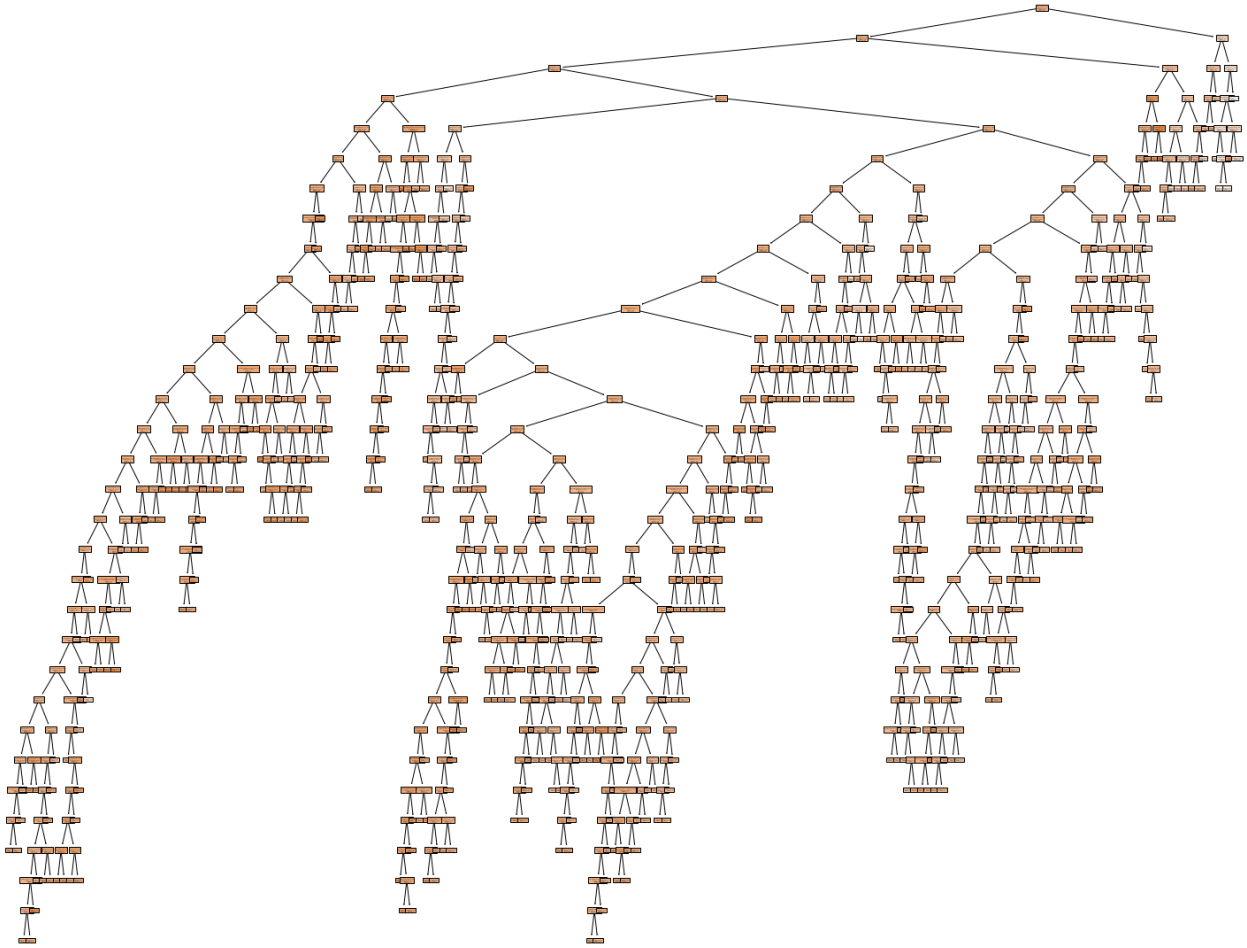

Exhibit 4: Optimized RFM

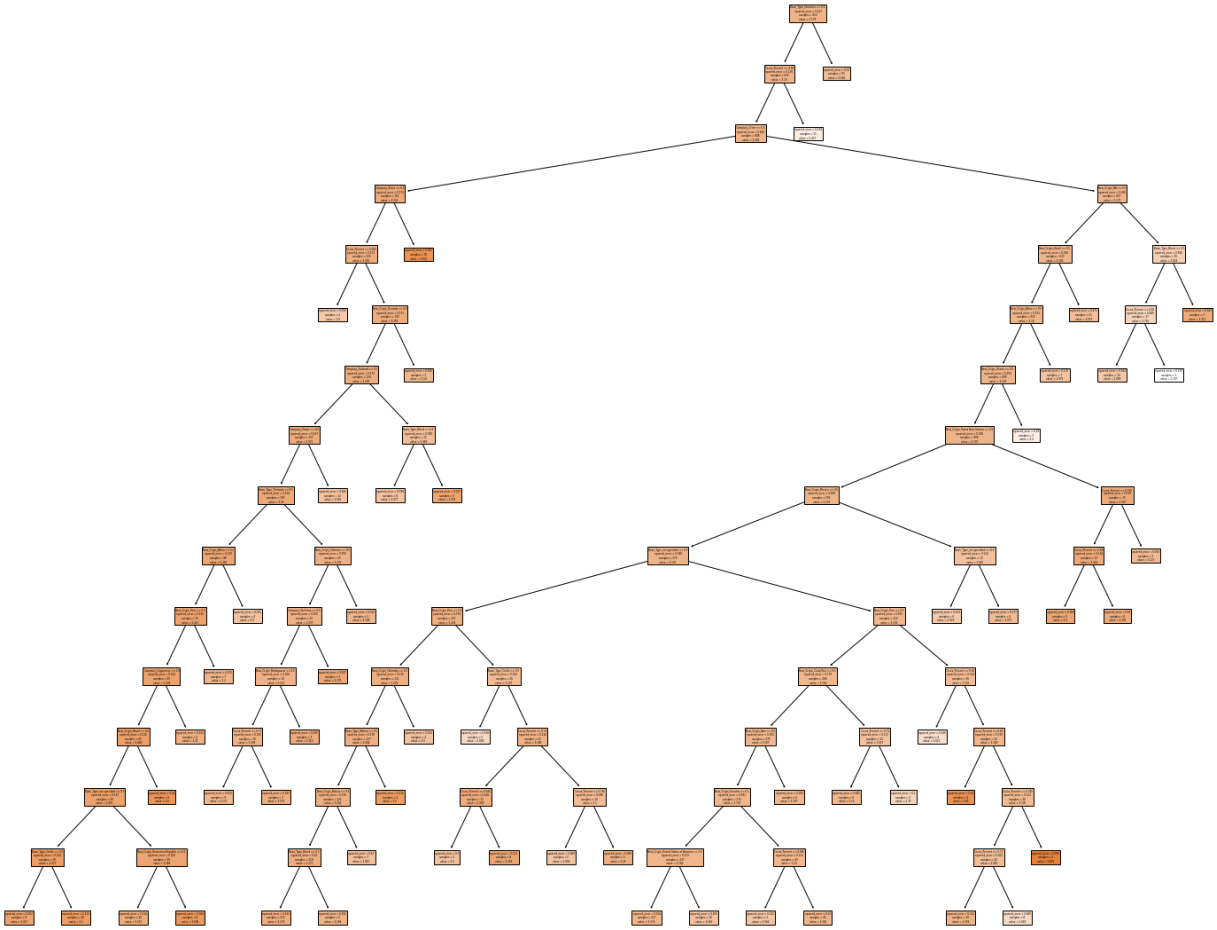

Exhibit 5: Model Evaluations

Model	Accuracy	RMSE	R2
Original DTM (Exhibit 1)	86.879832	0.474370	0.065212
Optimized DTM (Exhibit 2)	87.175782	0.466183	0.097200
Original RFM (Exhibit 3)	86.522252	0.488223	0.009819
Optimized RFM (Exhibit 4)	86.966983	0.465984	0.097972

Works Cited

31 current Chocolate Statistics (Chocolate Market Data 2022). Dame Cacao. (2022, December 5). Retrieved December 10, 2022, from https://damecacao.com/chocolate-statistics/#.

Bradford, K. | J. (2022, July 26). Chocolate the preferred Candy in america, poll finds. Food Beverage Insider. Retrieved December 10, 2022, from https://www.foodbeverageinsider.com/confectionery/chocolate-preferred-candy-america-poll-finds

Gokhale, C. (2018, March 4). Country to continent. Kaggle. Retrieved December 10, 2022, from https://www.kaggle.com/datasets/statchaitya/country-to-continent

Mayo, M. “Simplifying Decision Tree Interpretability with Python & Scikit-Learn.” *KDnuggets*, https://www.kdnuggets.com/simplifying-decision-tree-interpretability-toward-decision-rules-with-python-scikit-learn.html. Accessed 7 Dec. 2022.

Tatman, R. (2017, August 11). Chocolate bar ratings. Kaggle. Retrieved December 10, 2022, from https://www.kaggle.com/datasets/rtatman/chocolate-bar-ratings