For this analysis I used the data set that was originally created to understand what Halloween candy is preferred by most people. The goal is to try to predict if candy is chocolate or not based on other features included in the data set. Is the candy with nuts more likely to be chocolate candy? Or maybe it is more likely to be chocolate if it has caramel in it, nougat, or wafers? There is other information that can be uncovered with the help of this dataset: does chocolate candy contain more sugar than candy that is not chocolate. Is it more expensive? Finally, is chocolate candy more popular?

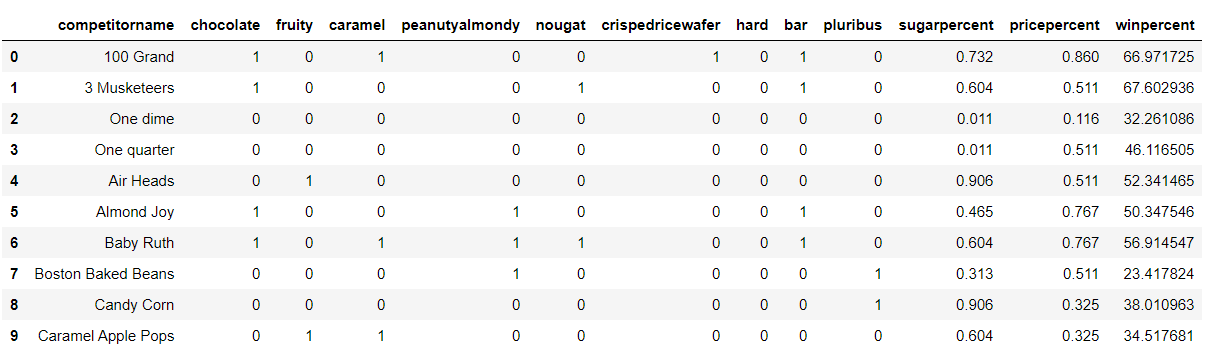
I downloaded the dataset from [Kaggle](https://www.kaggle.com/fivethirtyeight/the-ultimate-halloween-candy-power-ranking). The data was gathered with the help of a website that was created with the purpose of identifying most popular Halloween candy. Over 8000 Unique IP addresses participated in the vote of about 269000 matchups. I found this [article](https://fivethirtyeight.com/videos/the-ultimate-halloween-candy-power-ranking/) about the original analysis of Halloween candy pretty helpful and entertaining to read.

The variables used in the data set are:

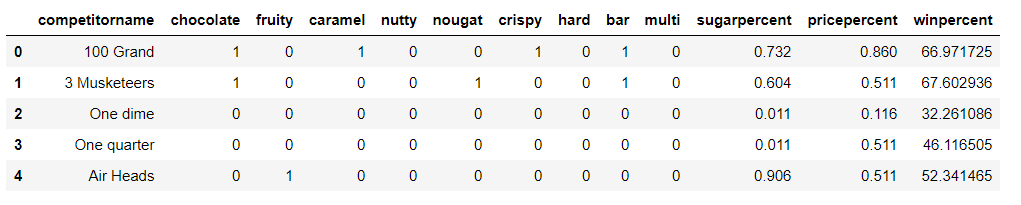
* chocolate: Does it have chocolate?
* fruity: Does it have any fruit flavor?
* caramel: Is there any caramel in it?
* peanutalmondy: Does it contain peanuts, almonds?
* nougat: Is there nougat in it?
* crispedricewafer: Is there any crisped rice, wafers, or a cookie component?
* hard: Is the candy hard?
* bar: Is it in the form of a bar?
* pluribus: Is there more than one candy in the box/bag?
* sugarpercent: The percentile of sugar.
* pricepercent: The item price percentile.
* winpercent: The win percentage according to all of the matchups.

**Step-by-step instructions for the graph analysis of the data to predict if the candy is chocolate. Steps that need to be performed:**

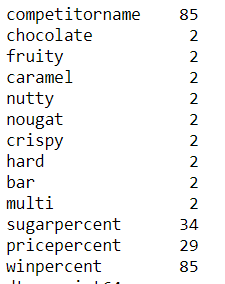
1. Load the data from the “candy-data.csv” file into a pandas DataFrame.
2. Display the dimensions of the table (it should be 85 observations of 13 variables).
3. Display the first 10 rows of the data to make sure it was loaded successfully and to check the column names and the observations.
   1. 9 variables are represented as a 1 or 0
   2. “chocolate” is the target variable



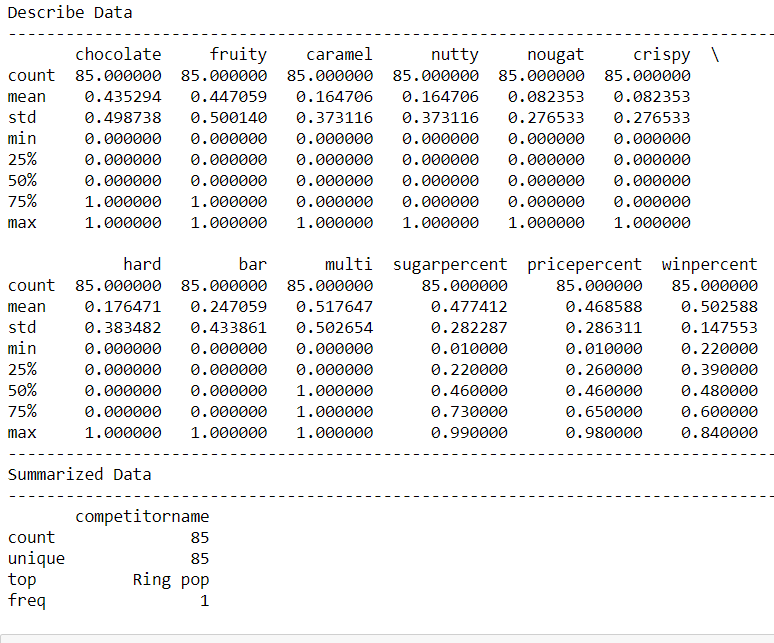
1. Some of the column names are very long and could be renamed to make it easier to work with in the future: from 'peanutyalmondy' to 'nutty; from pluribus to multi, from crispedricewafer to crispy.



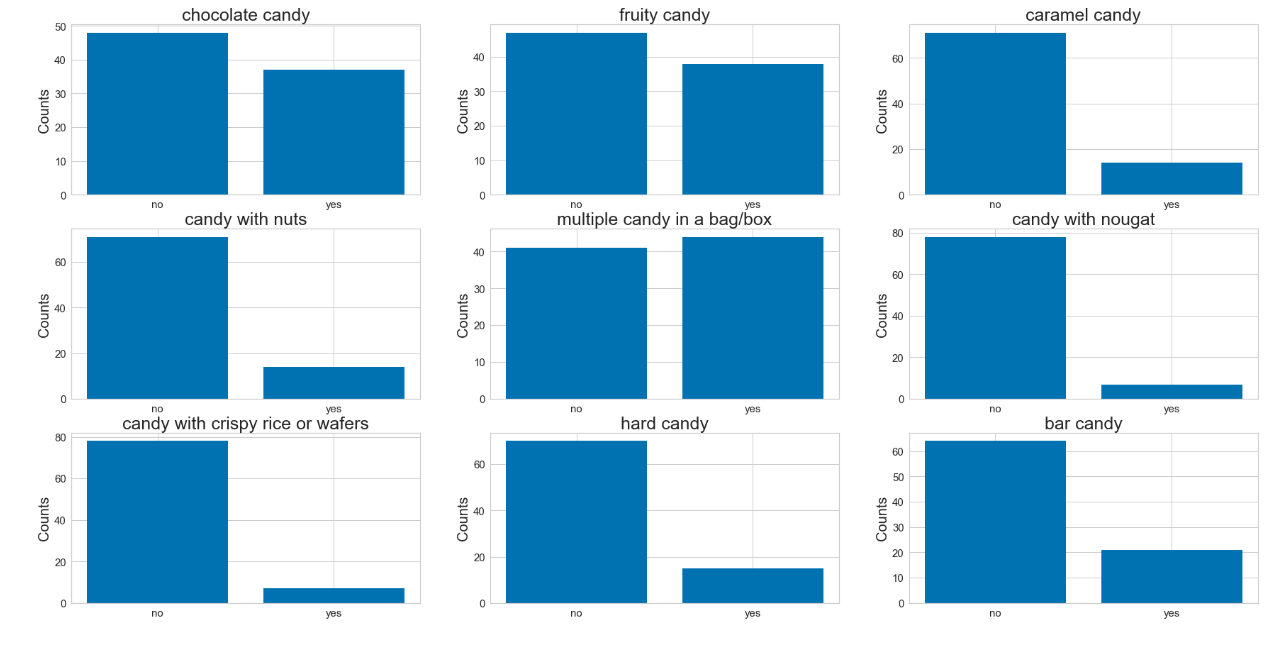
1. Check the number of unique values in each column. Pay attention to the variables with 2 unique values which are binary variables: for example 1-used for yes (chocolate) and 0- used for no (not chocolate).

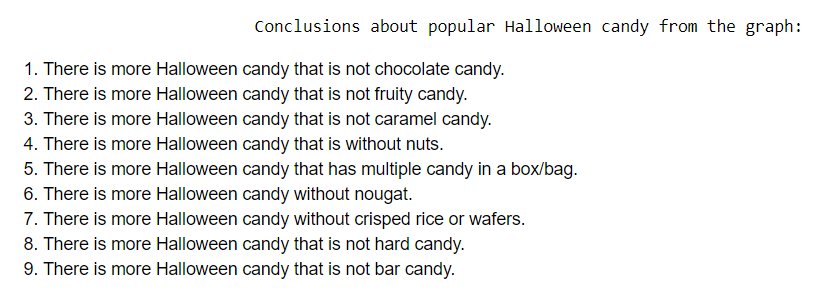


1. Check the summary information for this data:

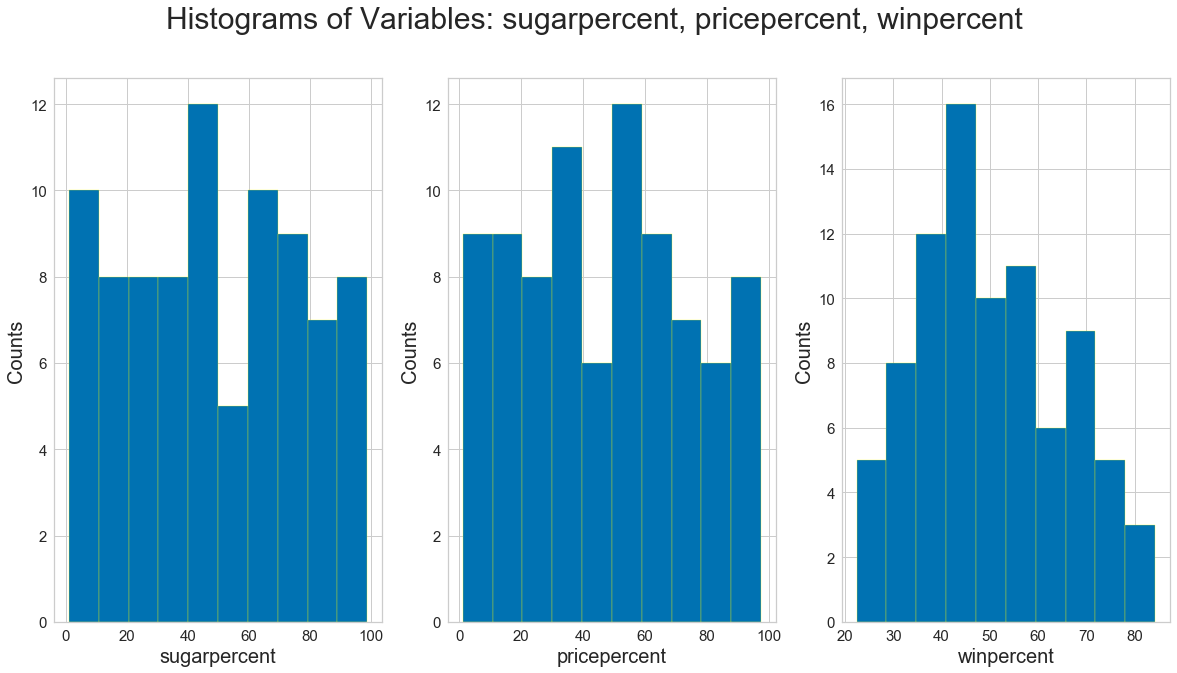


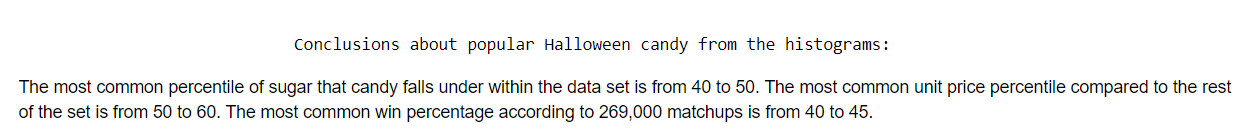
1. Make bar charts for the 9 binary variables:



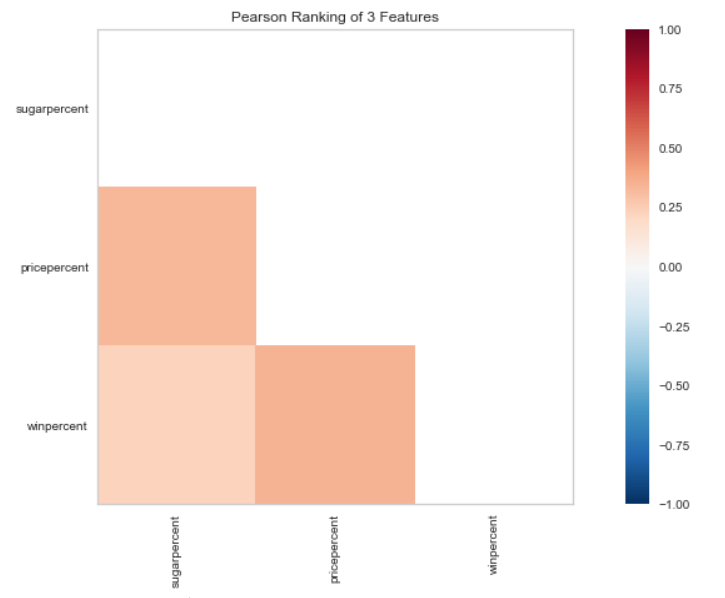


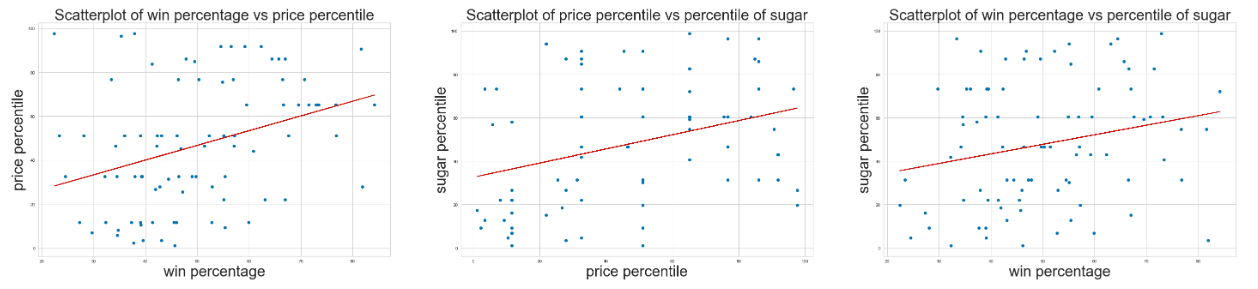
1. Make histograms of numerical variables that provide information about percentiles.

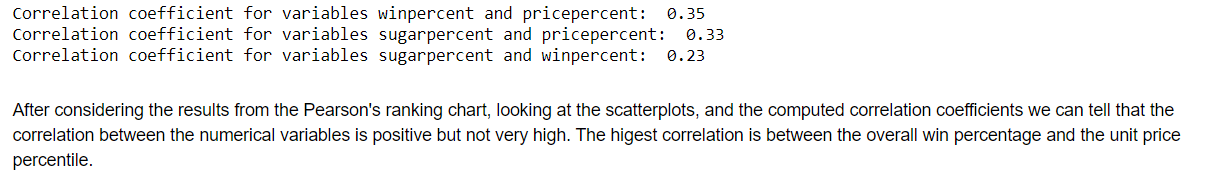




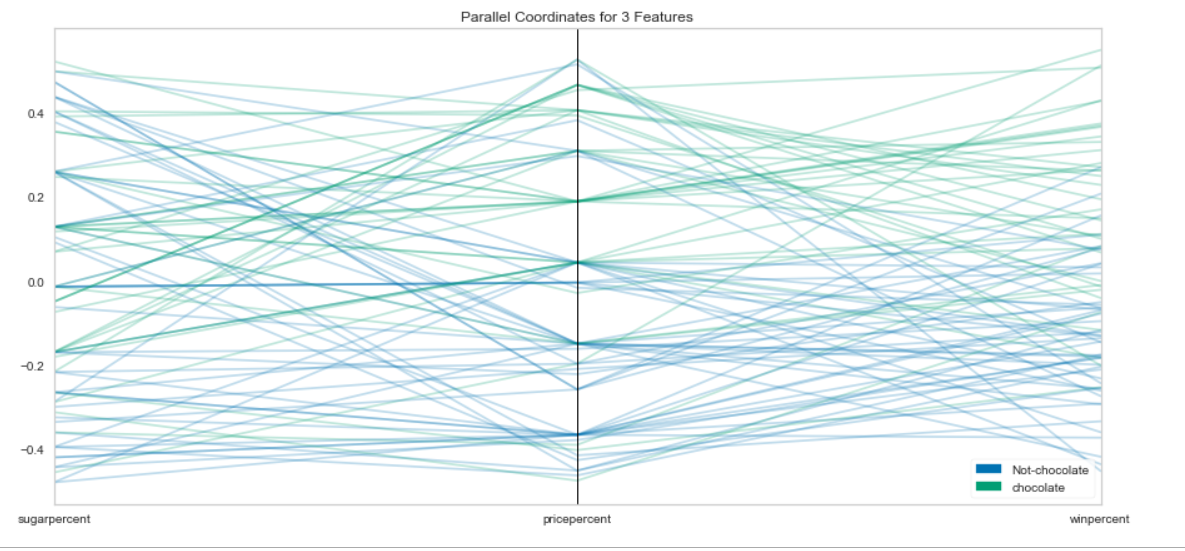
1. Make Pearson Ranking chart, create scatterplots to see the relationship between the variables and compute Pearson’s correlation coefficients:

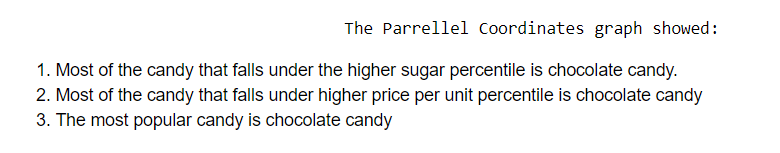




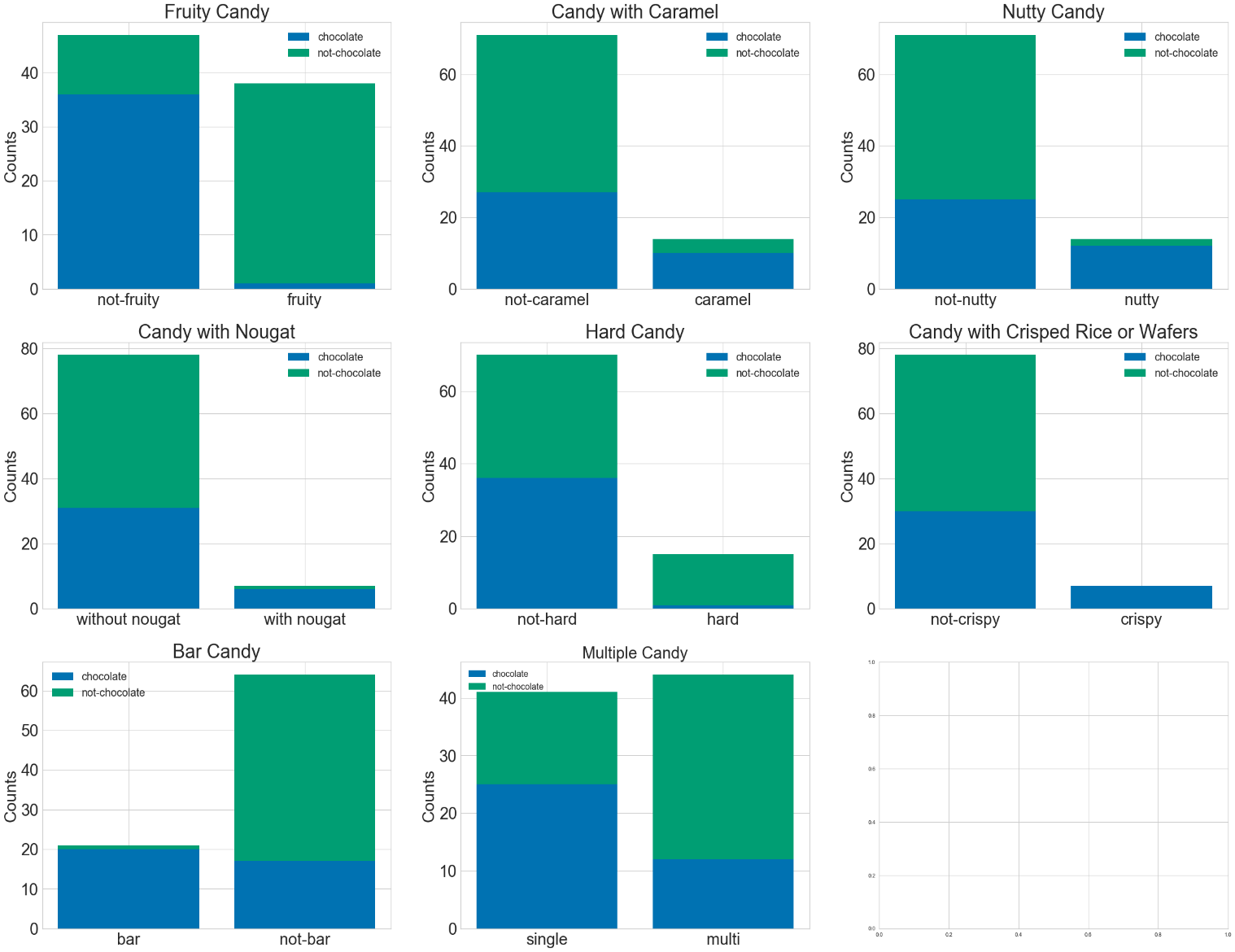


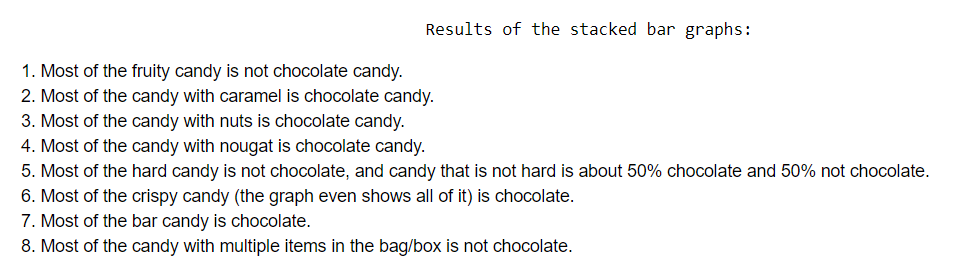
1. Create Parallel Coordinates graph for comparison of the distributions of numerical variables between chocolate candy and not chocolate candy.



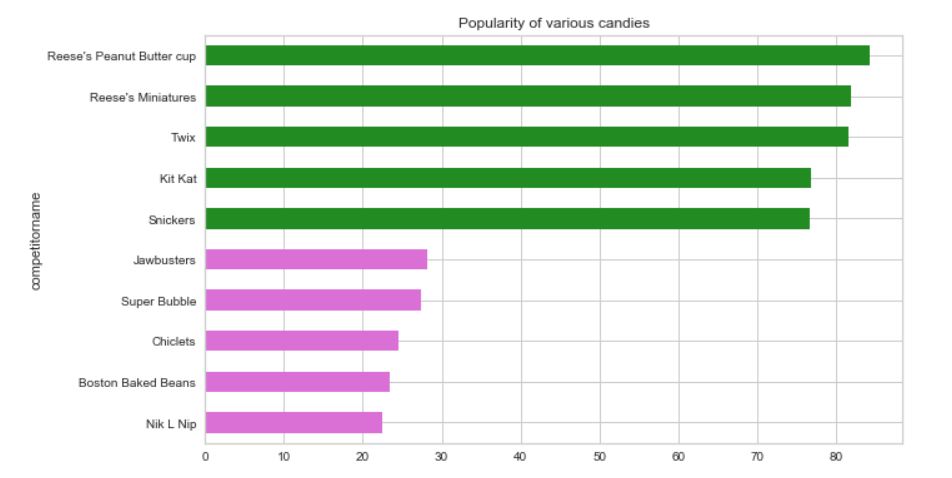


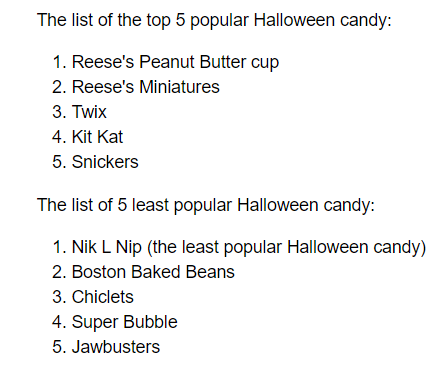
1. Use Stack Bar Charts to compare candy that is chocolate to that that is not chocolate based on the other variables.





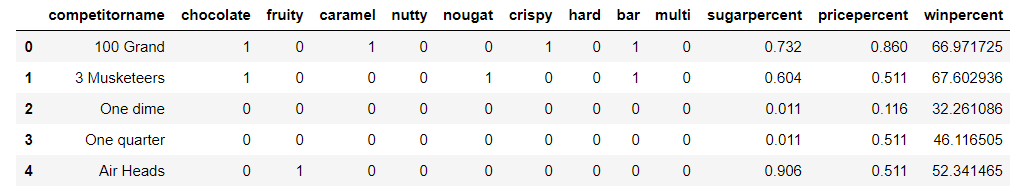
1. Create a bar graph of 5 most popular and 5 least popular candy.



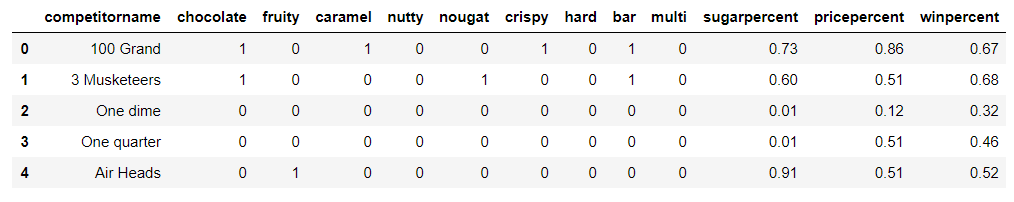


13. Use the same format for the continuous data. Since the other variables are dummy variables with 1 and 0 (we won’t need any transformations for them), using the numbers between 0 and 1 for continuous data that represents percentile or percentage in this data set would make sense.

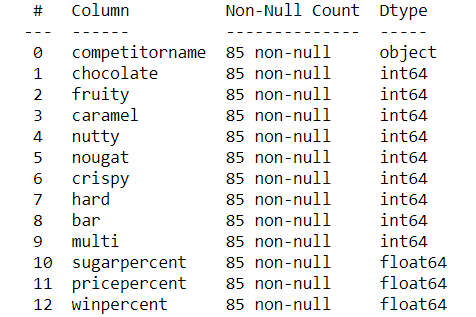
Original data:



**Normalized:**

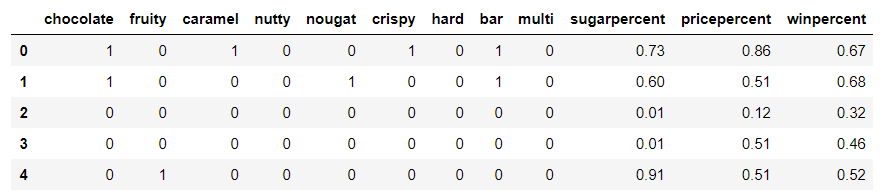


14. Check if there are any **missing values.**



**No missing values.**

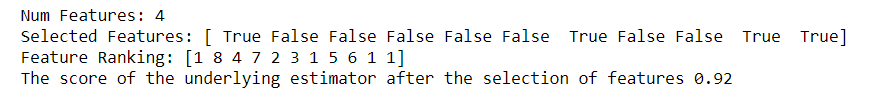
15. Removethe feature **‘competitorname’** since it is not providing any important information necessary for the classification analysis if candy is chocolate or not.



There is a total of **12 variables** now in the data frame.

16. Perform **feature selection** with the help of **Recursive Feature Elimination** (model-Logistic Regression).

The sample only has 85 observations. It is recommended to have at least from 10 to 15 observations per feature. Even though the number of features= 5 could be enough for this sample, to be on the safe side and not to overfit, use 4 since the sample is small.



17. Identify the names of the features selected.

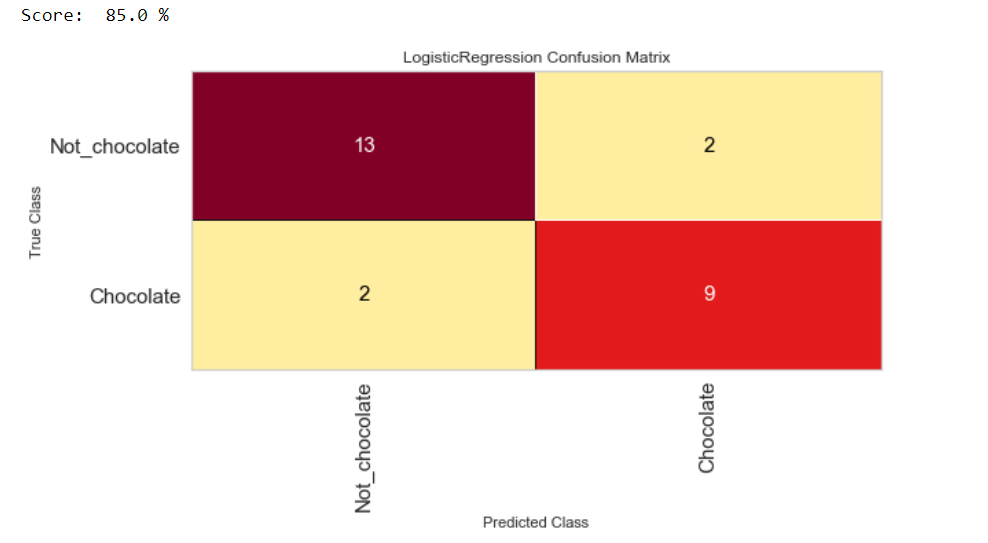
The selected features are **fruity, bar, pricepercent and winpercent.**

18. Create data set that could be used for training and validation data splitting.

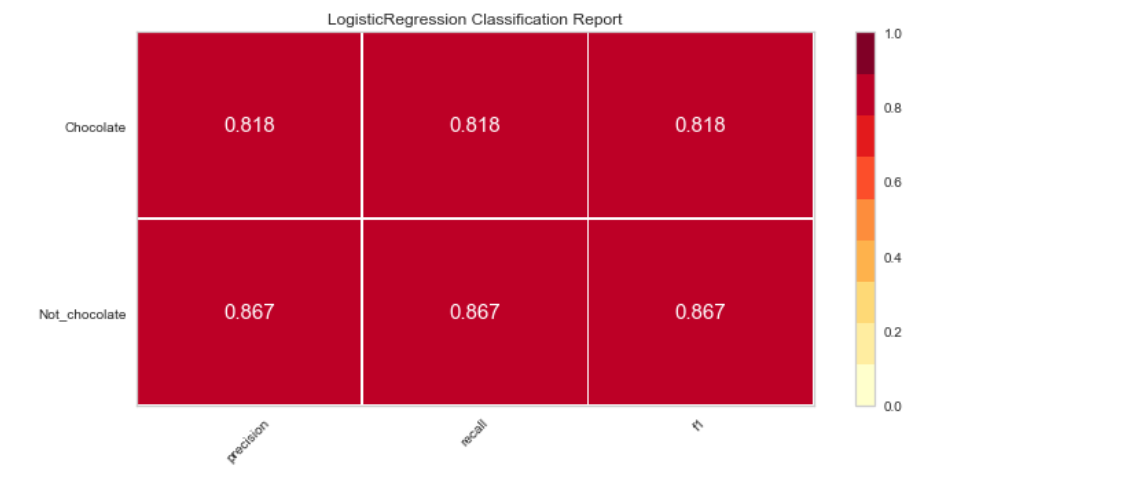
* + 1. Check if the target feature is balanced (you should get the ratio of 56/44)
    2. Split your data into two sets: Training and Testing. (My results were achieved with test\_size =0.3, random\_state=11)
    3. Check the number of chocolate and not\_chocolate observations for the resulting sets of data.

19. For feature selection we used RFE with Logistic Regression. Evaluate **Logistic Regression** model with the following metrics:

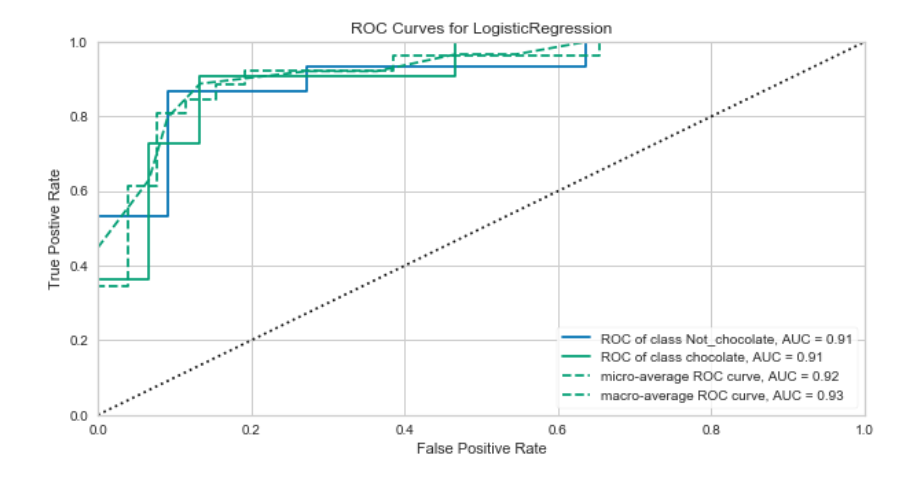
* + 1. Confusion Matrix (you should get 85%)



* + 1. Classification report: Precision, Recall & F1score. See image bellow



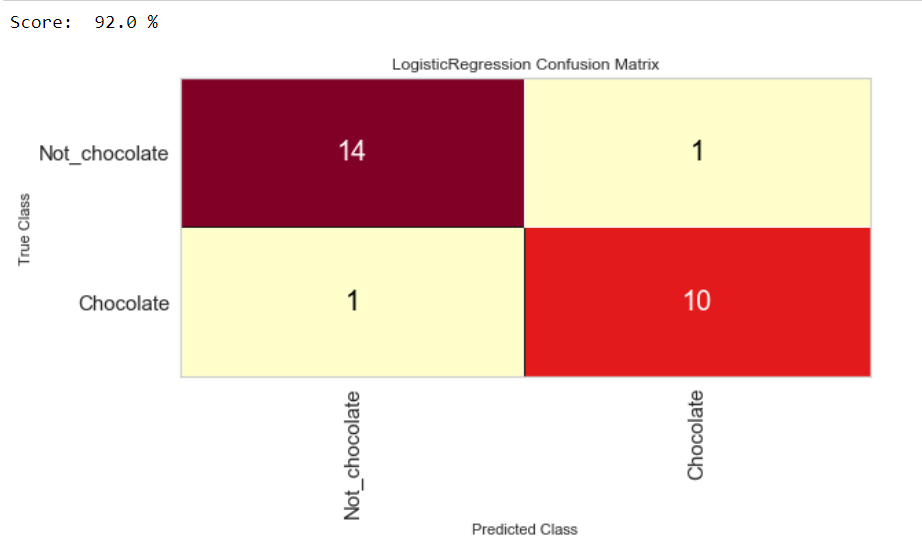
* + 1. ROC curve (the result is significantly above the dotted line)



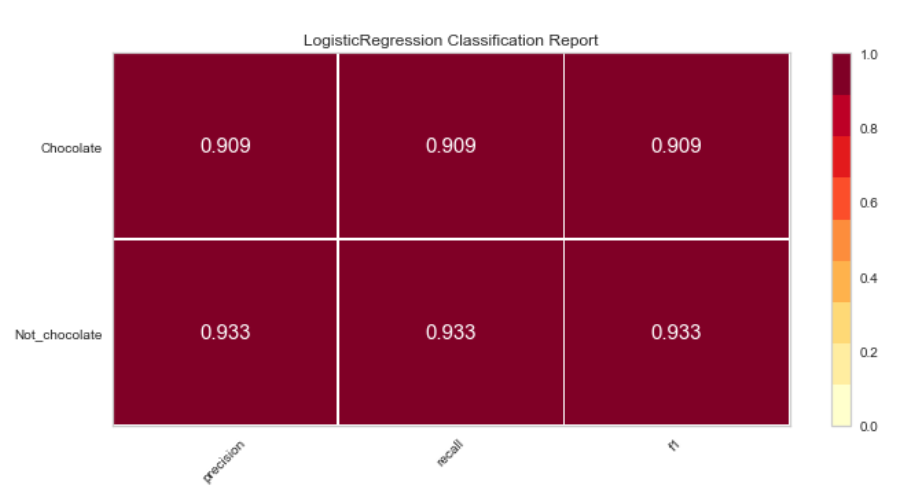
20. Use **GridSearchCV** to select optimal parameters and see if performance could be improved.

21. Evaluate the best model with the optimal parameters chosen after Grid search.

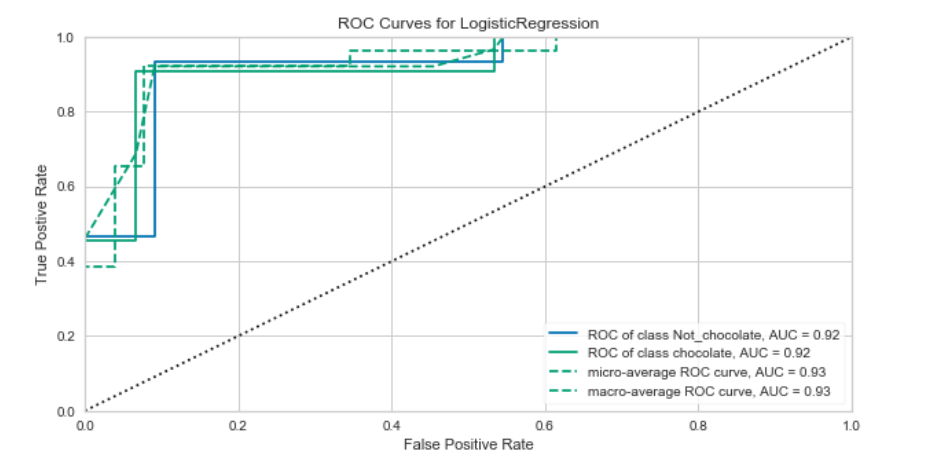
a. Confusion Matrix (you should get 92%)



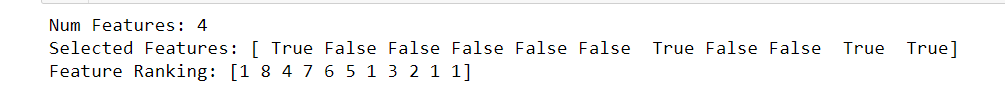
b.Classification report: Precision, Recall & F1score. See image bellow



c.ROC curve (the result is significantly above the dotted line (better than without hyperparameter optimization)



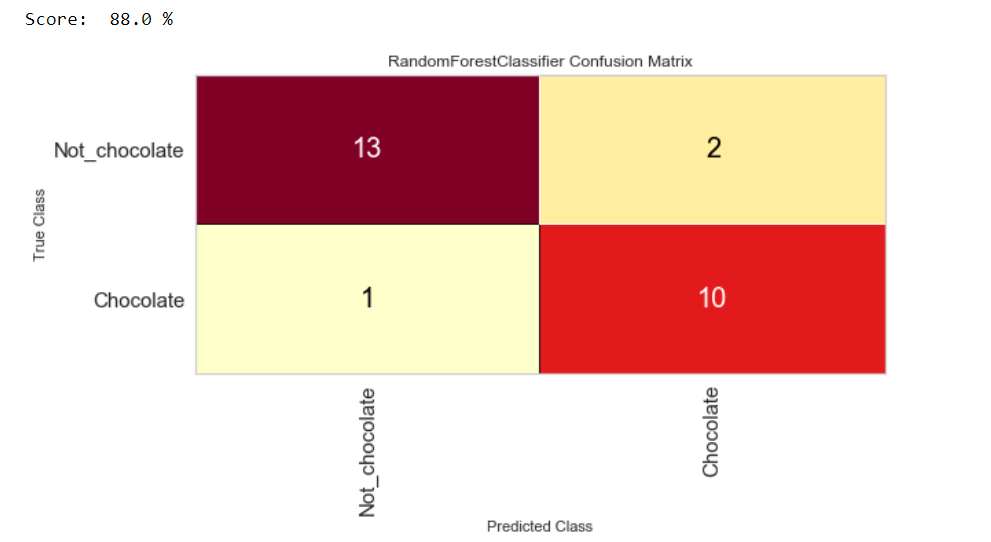
22. Perform **feature selection** with the help of **Recursive Feature Elimination** (model-Random Forest Classifier). The 4 most important features would be the same as for Logistic Regression.



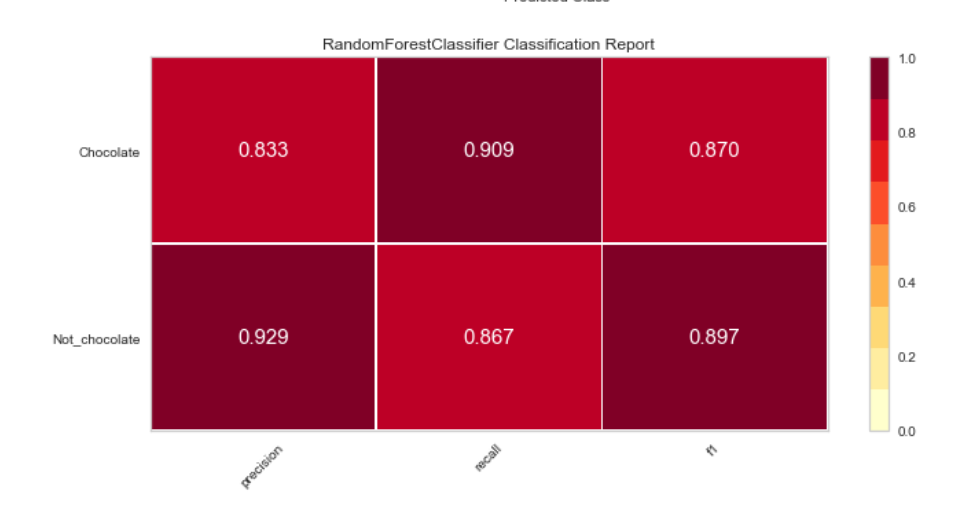
23. Run the **Random Forest Classifier** model

24. Evaluate the model:

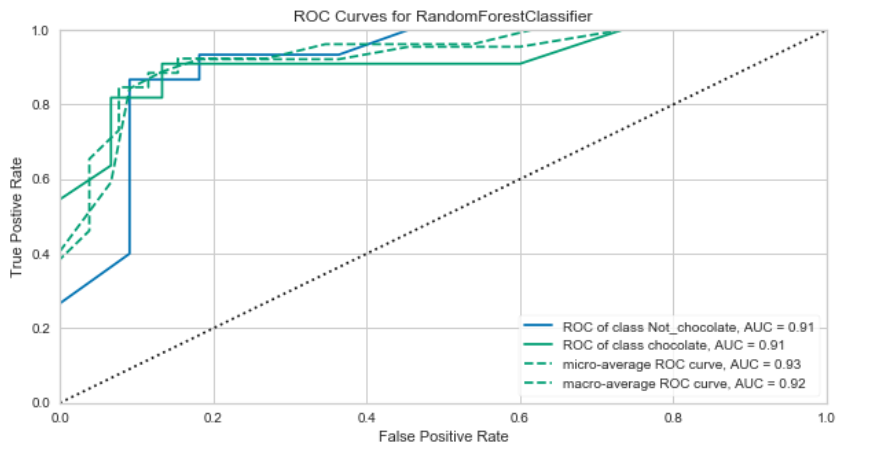
1. Confusion Matrix (you should get 88%)



b.Classification report: Precision, Recall & F1score. See image bellow



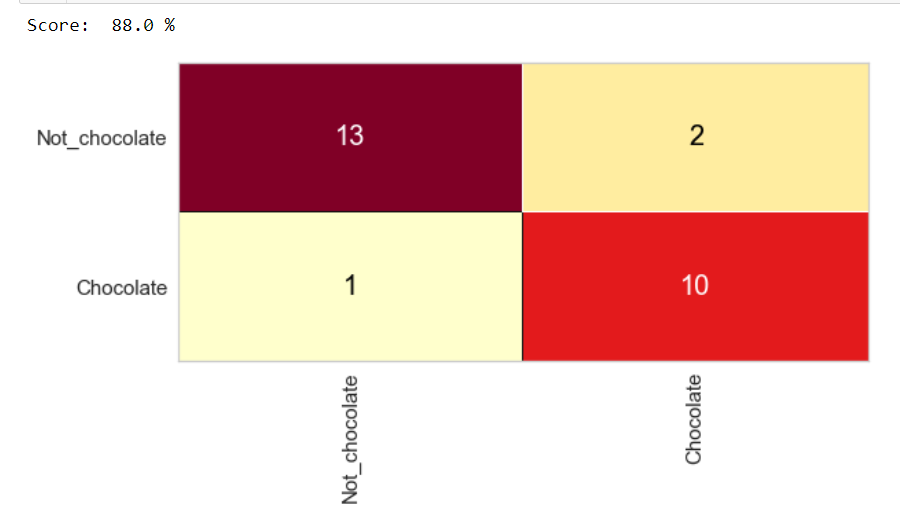
c.ROC curve (the result is significantly above the dotted line)



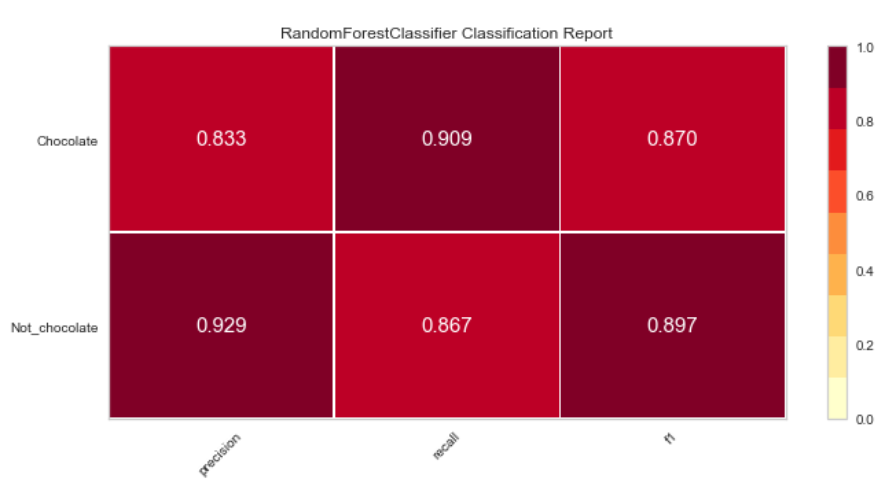
25. Use **GridSearchCV** to select optimal parameters for **Random Forest Classifier** and see if performance could be improved.

26. Evaluate the best model with the optimal parameters chosen after Grid search.

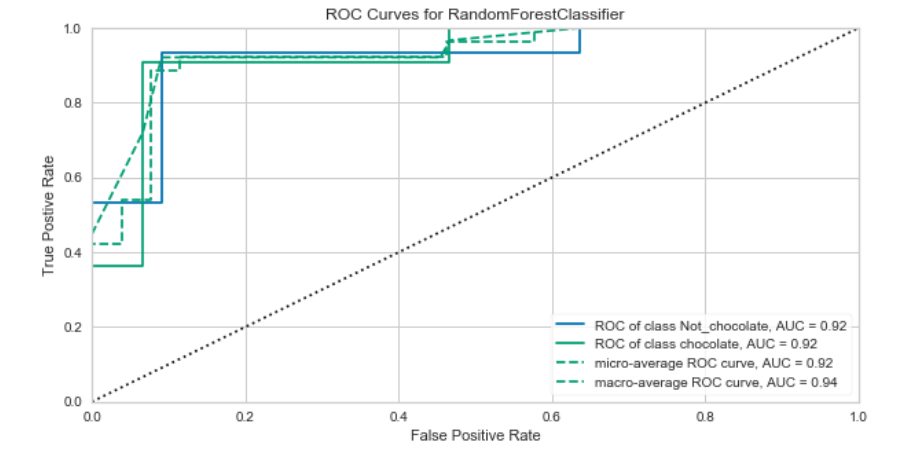
a. Confusion Matrix (you should get 88%)



b.Classification report: Precision, Recall & F1score. See image bellow



c.ROC curve (the result is significantly above the dotted line (better than without hyperparameter optimization). This is the only evaluation technique that showed better performance of a tuned RandomForest vs the one without custom selected hyperparameters.



27. The best results were demonstrated by the tuned logistic regression model, but all of the models produced good results. The data set originally did not have a large number of observations. More accurate results could be achieved after more data has been added.