Regression on Chocolate Bar Ratings Data Set

Patrick Brus

18.06.2021

Abstract

Chocolate is a candy almost everyone loves. One of the most famous forms in which chocolate is eaten is as a chocolate bar. However, not every chocolate bar tastes equal and is created equal. Therefore, a chocolate bar rating can be very helpful in order to find good chocolate bars. The chocolate bar ratings data set from Kaggle contains chocolate bars and their ratings. In this report, the data set is analyzed and feature engineering plus data cleaning is performed. Afterwards, several state-of-the-art regression techniques are evaluated and compared with the focus on prediction of the bar ratings over interpretation. The gradient boosting tree is used as final model due to its best performance and its parameters are optimized using a grid search. The final optimized gradient tree achieves a mean squared error of 0.2.

1 Introduction

Almost everyone loves chocolate bars. Even though chocolate is the biggest enemy of any diet, we can't stop eating it. However, the taste and quality of chocolate bars are very different. Some taste very good, while others do not really taste. Therefore, an evaluation of chocolate bars can be very helpful to make the right choice in advance. The Kaggle data set on [1] contains information about chocolate bars and their ratings. The machine learning goal is to predict the chocolate bar ratings given the information. The underlying machine learning task is therefore a regression. Before applying machine learning, the data set is explored and some feature engineering is applied in order to understand the data and to get the best performance of the machine learning algorithm. This report summarizes the findings after performing some Exploratory Data Analysis (EDA). In addition, feature engineering and pre-processing is applied in order to prepare the data set for machine learning. Afterwards, a training set and a testing set is created. The testing set should only be applied after the machine learning model is trained and optimized. The aim of the hold-out testing set is to check the performance of the model in the real world. As baseline model, a linear regression is trained. Afterwards, several state-of-the-art machine learning regression techniques are trained and compared to each other. The best performing is then optimized and used as final model. The code can be found on my Github page [1] in Jupyter notebooks

2 Attribute Information

This chapter gives a quick overview of all available features and a short description of them. Table 1 shows all feature names and a short description of each feature.

[1] link to Github: `https://github.com/patrickbrus/IBM_Machine_Learning_Professional/tree/master/Regression`

Feature Name	Description
Company	Name of the company manufacturing the bar.
Specific Bean Origin or Bar Name	The specific geo-region of origin for the bar.
REF	A value linked to when the review was entered in the database. Higher = more recent.
ReviewDate	Date of publication of the review.
CocoaPercent	Cocoa percentage (darkness) of the chocolate bar being reviewed.
CompanyLocation	Manufacturer base country.
Rating	Expert rating for the bar.
BeanType	The variety (breed) of bean used, if provided.
Broad BeanOrigin	The broad geo-region of origin for the bean.

Table 1: An overview of all features available in the chocolate bar ratings data set.

2.1 Ratings

In the description of the data set it is stated, that each chocolate is evaluated using a combination of objective qualities and subjective interpretation. The rating itself is done using a chocolate bar of one batch, with batch numbers, vintages and review dates being included in the data set. Table 2 shows an overview of the different rating levels and their meaning.

Rating	Meaning
5	Elite (Transcending beyond the ordinary limits)
4	Premium (Superior flavor development, character and style)
3	Satisfactory(3.0) to praiseworthy(3.75) (well made with special qualities)
2	Disappointing (Passable but contains at least one significant flaw)
1	Unpleasant (mostly unpalatable)

Table 2: An overview of all rating levels and their meaning.

3 Exploratory Data Analysis, Feature Engineering and Data Cleaning

This chapter describes the process of analyzing the data, cleaning it and creating new features. These steps are combined in one chapter, because they often go hand in hand. As first step, the pandas data frame head function is used to get an initial view into the data. The column names are not well formatted and are containing newline characters. So as first step, the column names are renamed (Figure 1). Secondly, the pandas data frame function $info()$ is used in order to quickly check which data types are available and if data is missing. The columns $Company$, $Spec\_Bean\_Origin\_or\_Bar\_Name$, $Cocoa\_Percent$, $Company\_Location$, $Bean\_Type$ and $Broad\_Bean\_Origin$ are categorical features and have to be transformed. When looking at the missing values, then only the features $Broad\_Bean\_Origin$ and $Bean\_Type$ are containing one missing value out of 1795 total samples. However, when looking at the data frame head, the first five columns of feature $Bean\_Type$ are empty and should be therefore count as missing value.

Therefore, the first entry of $Bean\_Type$ is fetched in order to check its value and to use this for replacing these values with NaN. After replacing the empty values of $Bean\_Type$ with NaN, the $info()$ function is called again and shows now that the feature contains 891 missing values, which is almost 50% of the total number of samples. As a next step, the feature $Cocoa\_Percent$ is transformed by removing the trailing percentage sign and casting the data type to a numerical data type.

Afterwards, a list of numerical and categorical features is created and a histogram is plotted for each numerical feature (Figure 2). As one can see, the period of reviews in the data set goes from 2006 to 2017. The distribution of the feature $Cocoa\_Percent$ seems to be a little bit skewed and can be transformed later. The data set does not contain any ratings at five, so there are no "Elite" chocolate bars. Most of the ratings are between three and four, meaning that most chocolate bars are "Satisfactory".

After closer examination of the categorical features, it can be seen that they contain a lot of categories and also partially single examples for some categories, which is not very helpful for the machine learning algorithm. Therefore, these are first to be transformed.

The first transformed categorical feature is the $Bean\_Type$. Figure 3 shows the histogram of all available bean types. The most frequent bean types are Criollo, Trinitario, Forsastero. The most of the other bean types are simply a mixture of these bean types. Blend also appears often, but only means that this bean type is a mixture of other bean types. Some others and less common bean types are Beniano, Blend, Matina, EET, Nacional, Amazon. Therefore, one column is created containing all the here mentioned bean types and each sample is then checked for the presence of these categories. If any of the categories occur, then a one is added to the category column for that sample. If a sample contains multiple bean types, multiple ones are set. Thus, the intermixed bean types can be mapped to categorical features. It could be also useful know whether a chocolate bar contains a mixture of bean types or not. Maybe this already has an influence on the final rating. Therefore, a new feature called Num_Beans is added. This feature contains the number of different bean types a chocolate bar contains. Finally, a feature Is_Blend is added to hold whether a chocolate bar contains a mixture of beans or not and the bean types with type Blend are also getting a True for this feature.

The next feature that is transformed is the $Spec\_Bean\_Origin\_or\_Bar\_Name$. There are 1039 different categories, which is a lot. There are also again a lot of categories only containing one value, which would not deliver any useful information for the machine learning algorithm. A stemmer is used in order to map words with the same meaning together. This already reduces the number of categories to 682 different categories. The single occurrences categories are mapped to the category "Other", which reduces the size of different categories to 207.

As a next feature, the broad bean origin ($Broad\_Bean\_Origin$) is closer examined. In total, there are 100 different categories and some categories are again only a comma separated list of other more common categories. In addition, some countries are written differently sometimes (i.e. Dominican Republic, D.R. D. Republic, Dominican R.). Therefore, regular expressions are used in order to ensure that all different spellings are mapped to the same country. The broad bean origin feature also contains almost 50% missing values. These missing values are replaced by the category $Unknown\_Bean\_Origin$. The comma separated list is split in order to get a list of categories, and then the same approach as for feature $Bean\_Type$ is applied. After all the transformations, only 51 different categories are left. Figure 4 shows the final histogram after all transformations on that

```
df = df.rename(columns={"Company\xa0\n(Maker-if known)": "Company",
                        "Specific Bean Origin\nor Bar Name": "Spec_Bean_Origin_or_Bar_Name",
                        "Review\nDate": "Review_Date",
                        "Cocoa\nPercent": "Cocoa_Percent",
                        "Company\nLocation": "Company_Location",
                        "Bean\nType": "Bean_Type",
                        "Broad Bean\nOrigin": "Broad_Bean_Origin"
                       })
```

Figure 1: Python code for renaming the column names.

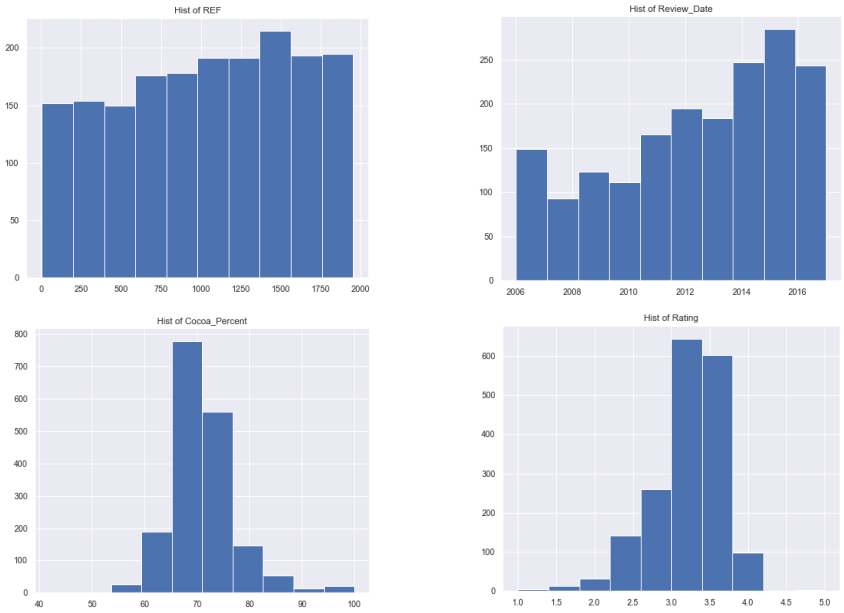

Figure 2: Histograms of numerical features of chocolate bar ratings data set.

feature are applied.

The feature *Company_Location* is the last transformed categorical feature. Here, only the single occurrences of the company locations are again replaced with the category "Other". Finally, all the categorical columns left are transformed using pandas *get_dummies*() function.

As a last transformation, the skewed feature of cocoa percent is transformed using a log transformation. With this, the skewness is reduced from 1.06 to 0.3. Figure 5 shows the histograms before and after the applied transformation. As one can see, the transformed histogram looks way less skewed then the one before the transformation. As a last step, the data is split into a training and a hold out test set by using 20% of the data for testing.

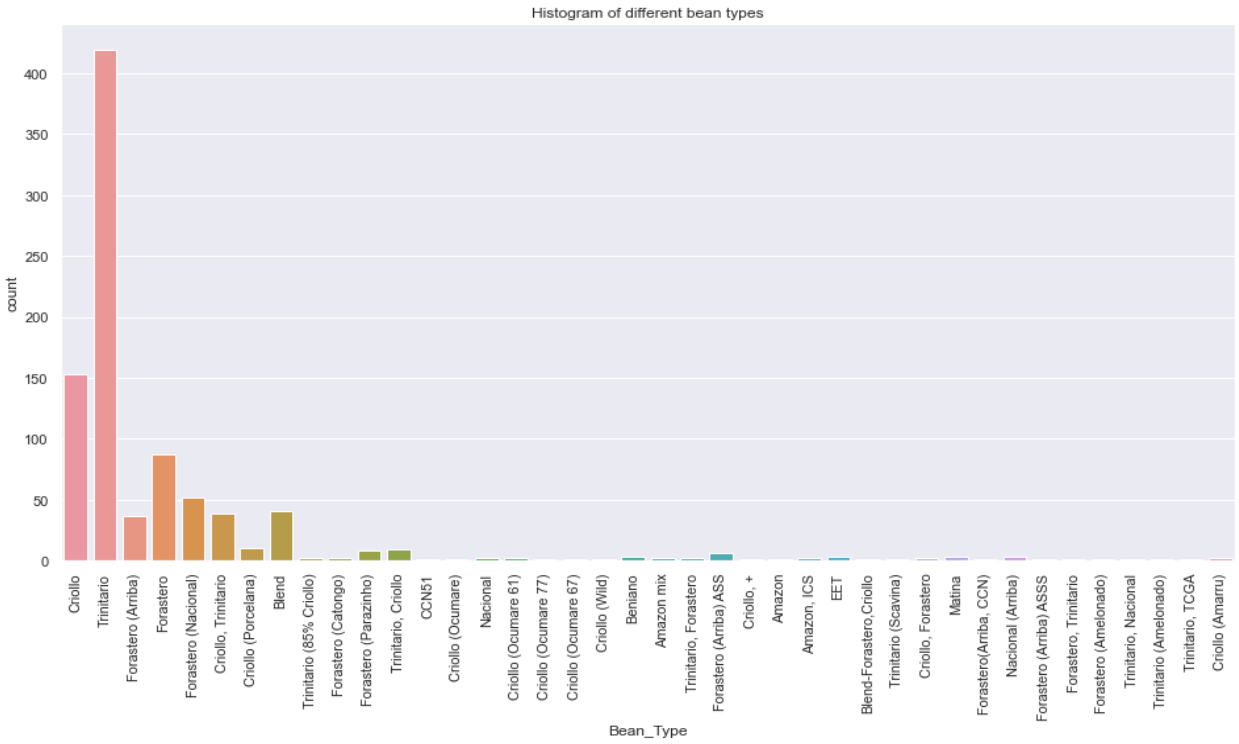

Figure 3: Histogram of all the available bean types after the transformations.

4 Methodology

As stated in the introduction, the underlying machine learning task is a regression task. Therefore, linear regression is trained and then used as initial baseline. Before training the linear regression model, the data is scaled in order to ensure that all features are at the same scale. Thereby it is important that the scaler is only trained using the training data and not the test data. Therefore, a scikit-learn pipeline is created with the StandardScaler and the LinearRegression functions. The cross validation prediction function is then used using a five-fold cross validation. The linear regression model achieves a mean squared error of $1.4e26$ and a larger negative r-squared score. This clearly shows that the underlying problem can not be solved by using a linear regression.

As next step, the following state-of-the-art regression machine learning algorithms are applied:

1. Ridge Regression
2. Decision Tree Regressor
3. Random Forest Regressor
4. Gradient Boosting Regressor

Each algorithm is again trained by using a five-fold cross validation and by first scaling the input data. Table 3 shows all mean squared errors of the different algorithms. As one can see, the gradient boosting algorithm achieves the best results and is therefore used for the final model. As

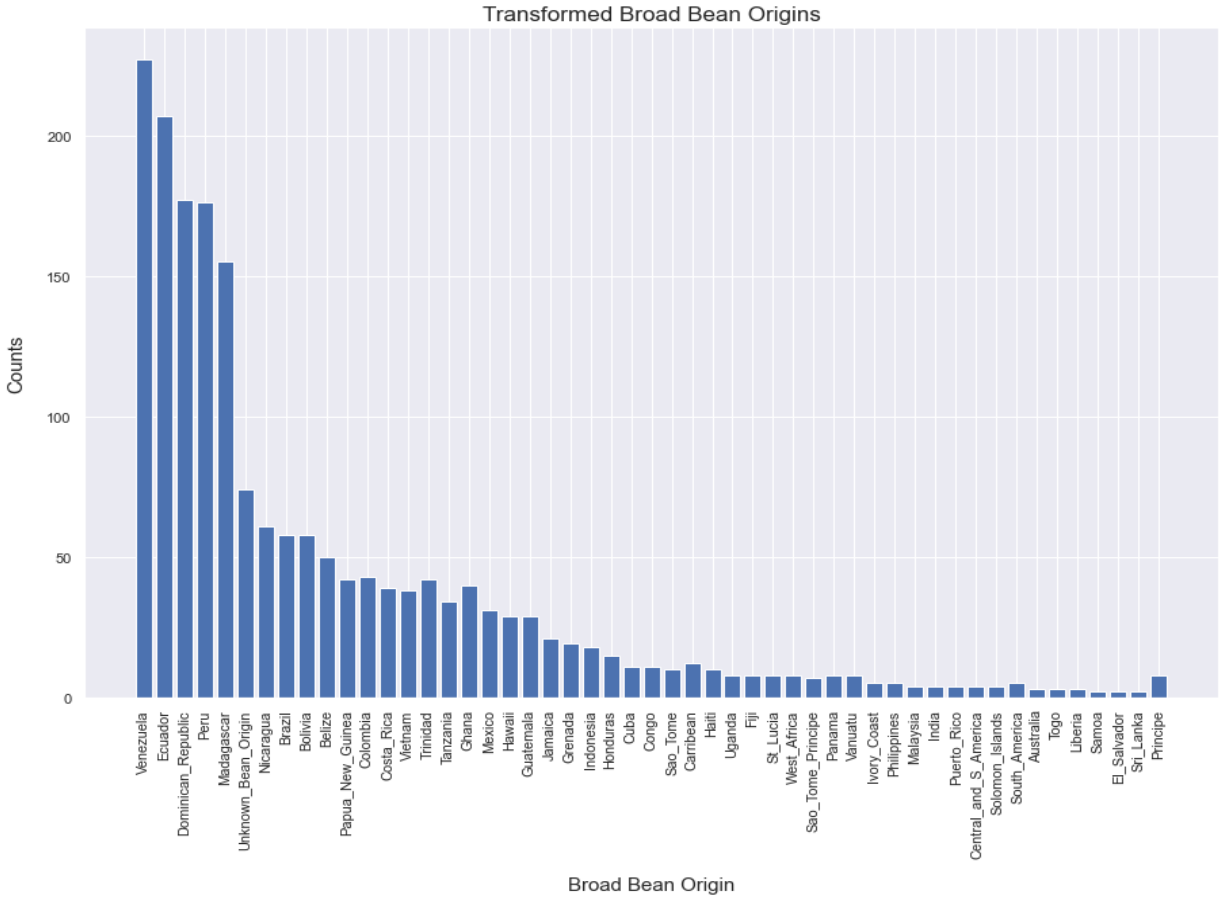

Figure 4: Histogram of all the available broad bean origins after the transformations.

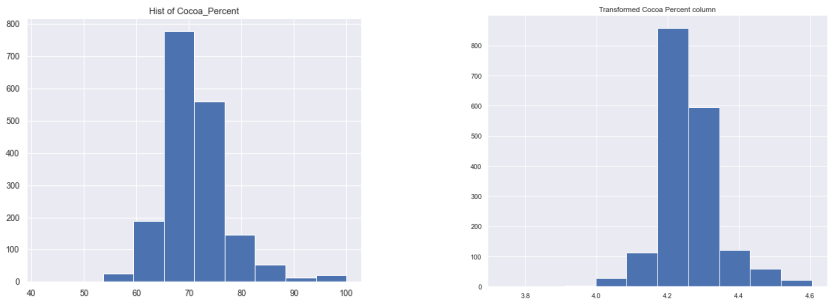

Figure 5: Histograms of numerical feature cocoa percent before the log transformation (left) and after the transformation (right).

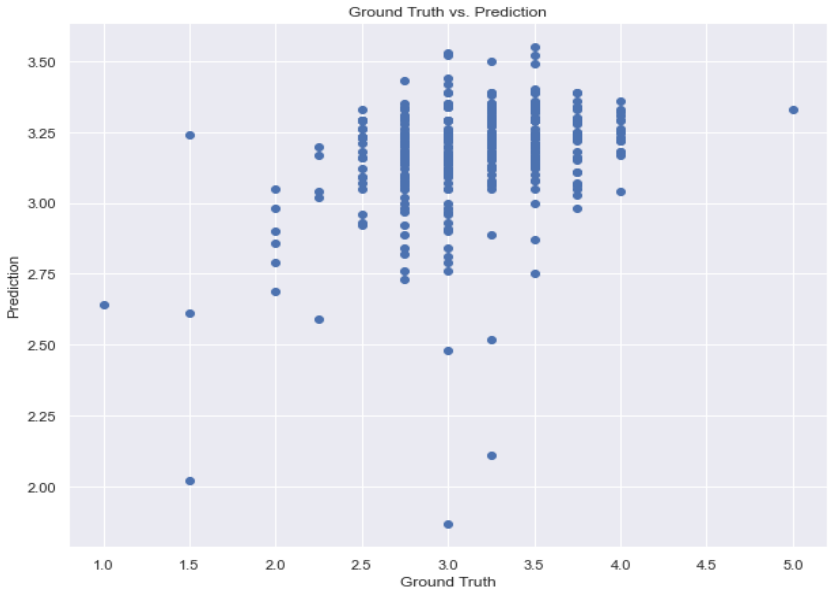

Figure 6: Scatter plot of ground truth vs. predicted ratings on hold-out test set.

a last step, the hyper-parameters of the gradient booster are optimized by applying grid search. The full optimized gradient boosting algorithm achieves an mean squared error of 0.131.

Algorithm	**Mean Squared Error**
Ridge Regression	0.250
Decision Tree Regressor	0.329
Random Forest Regressor	0.199
Gradient Boosting Regressor	0.196
1	Unpleasant (mostly unpalatable)

Table 3: Training results after the cross validation of the different machine learning algorithms.

5 Results

The final optimized gradient booster is then used to predict the ratings on the hold-out test set in order to check the performance on the "real world" data. The mean squared error on this is 0.199, which is quite low and not a lot above the training error. This indicates that there is no over-fitting. Figure 6 shows a scatter plot with the ground truth data on the x-axis and the predicted data on the y-axis. As one can see, the model predicts also ratings in between of the possible rating values. As one possible future step, the ratings could be mapped to the according real ratings, if wished. Figure 7 shows the ten most common features and their importance on the ratings. The most important feature is the cocoa percent, followed by the value linked to when a review was entered in the database. The added feature *Is_Blend* seems also to be very important, which is very nice because that means that it was a good idea to add this feature.

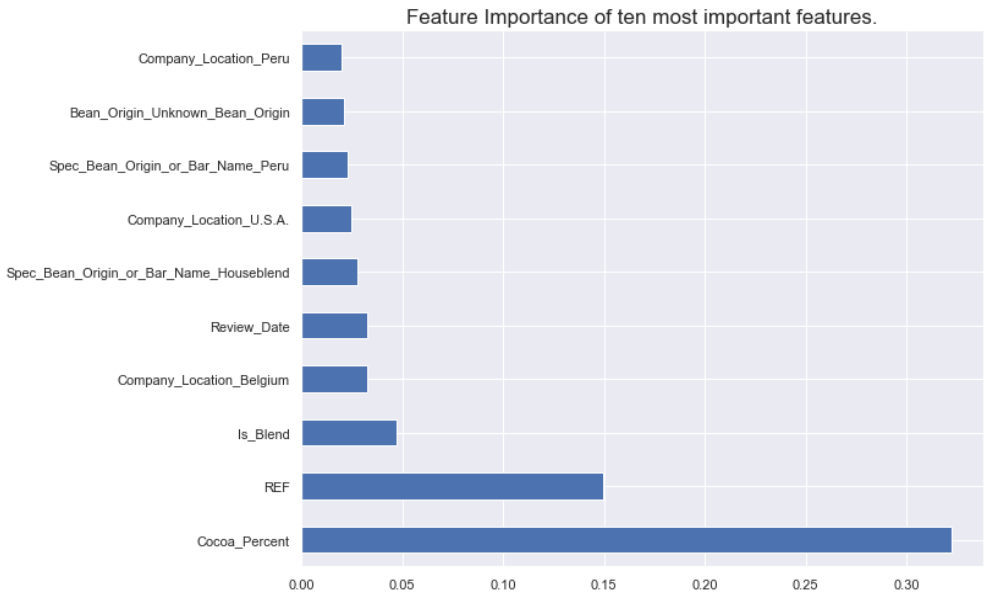

Figure 7: Bar plot of the ten most important features and their weight.

6 Future Work

This section briefly discusses some future work that could be done. As one step, the feature *Company_Name* could be better processed, because currently it is only dropped from the data frame. In addition, a neural network could be added as additional machine learning algorithm. Maybe the neural network can achieve an even better performance. Furthermore, the predicted ratings could be mapped to real ratings, which are available with a resolution of 0.25.

References

[1] Rachael Tatman. *Chocolate Bar Ratings*. 2017. URL: `https://www.kaggle.com/rtatman/chocolate-bar-ratings` (visited on 06/18/2021).