ML Engineer Nanodegree Capstone Proposal

Lara Lu

February 12, 2018

1 Domain Background

According to Mordor Intelligence, the wine market is value at 287.39 billion USD in 2016, and is forecasted to reach 402 billion by 2022 [1]. In the US in 2016, 806.45 million gallons of wine was produced [2]. For a beginner, choosing a good bottle of wine can be difficult as there are usually too many options, whether in a liquor store, the wine aisle in a supermarket or online. Many people like me tend to rely on wine ratings from trusted sources, such as WineEnthusiast, to help make a more informed purchase decision. This process works great except for new wines on the market which don't have many reviews yet. In 2017 there are almost 400 new wineries in the US alone[3] - it is hard to tell from the myriad of new bottles which ones are gems and which ones to avoid. It would be ideal if there is a way to predict the score of a wine from the information we already know. Currently there is a dataset on Kaggle that is scraped from WineEnthusiatic.com. I will be utilizing this dataset, as explained in section 3, to get better understanding of the relationship between the wine's attributes and its score.

2 Problem Statement

Is it possible to predict the rating of a new wine based on what we know about it? In this project, I would like to explore the rating data collected from WineEnthusiatic.com, and try to use information such as the description, designation, price, and winery to predict the its score.

3 Datasets and Inputs

The dataset I'm going to use is the one on Kaggle, which is scraped from WineEnthusiast during the week of June 15th, 2017. The dataset contains 150,930 entries, each for a different wine, with columns regarding the wine's production and market information, such as country, description, designation, price, region and winery, as well as the points given by a contributing editor of the wine magazine. I will be use all columns but the points as raw features and points as the label. Features eventually need to be in vector form.

The simplest feature is price, as it is already numerical, and I will keep it as is.

Country, region, province and winery are categorical values, and are usually preprocessed through one-hot encoding. However, because of the myriad of regions and wineries, using one hot encoding for columns with too many values can explode the dimension of input features and make it difficult for the model to converge because of the curse of dimensionally. In the project, I will examine the number of values in a column and only use the ones where majority of the values fall into a small number of buckets.

Description is a potentially long free text corpus and requires more complex pre-processing. First we need to tokenize the strings. A naive method to follow is bag of words, which assigns each string an index and generates a count as the new feature vector. The downside is long corpuses will have higher counts than shorter ones. Term frequency (tf) addresses this issue by using frequency, which is the string count divided by the length of the corpus. One issue remains: usually, words that are common to many corpuses (the, a, is, etc) carry very little meaning. In order to discount common terms, we can use tf-idf which multiplies term frequency by an idf component which reflects how uncommon a string is across all documents, and therefore giving uncommon terms more weight than common ones.

4 Solution Statement

This is a supervised machine learning problem as we are using a number of input features to predict an output label. Furthermore, because the output is numeric and continuous, it is a regression instead of classification problem.

A linear regression is an easy starting point to regression but unfortunately is not suitable here. Our feature is a long vector as a result of vectorizing description (and other fields), which likely has complex relationship with the final wine score, and the power of linear function is unlikely to capture the non-linearity of the relationship.

Random forest regressor, an ensemble method that combines multiple decision trees, is usually more powerful than linear regressor, as it is generally able to model complex non-linear relationships. However, it is not a good bet in our case either because our feature vector is going to be sparsen. Selections of node splits of the underlying decision trees are going to waste most of the model's insights on zero-only entries.

A kernal-based regressor might work better. I would like to try using a stochastic gradient descent regressor. The loss function is ordinary least squares fit, which is a linear function aiming to minimizing the sum of squared errors. The error will be used to back-propogate and update parameters scaled by the learning rate. Stochastic is as opposed to batched approach. In stochastic gradient descent, we update parameters for each training data point individually.

As mentioned on the sklearn documentation, SGD has been able to show success on large and sparse data, which matches the characteristics of our use case.

5 Benchmark Model

I have not found any existing solution to the regression problem I am trying to solve for the wine dataset online. For the sake of comparison, I will use a use a linear regression model on the same features as benchmark. Linear regression tries to find a weight for each feature and an intercept so that the residual sum of squared error is minimized.

6 Evaluation Metrics

A common metric for regression model is coefficient of determination or R2. R2 measures the proportion of the variance in the dependent variable which is predictable from the independent variable. 1.0 means perfect correlation, a constant model always outputting the same output will score 0, and it is possible to get a negative score. The higher the R2 score, the better the model is.

7 Project Design

I plan to first use pandas and matplotlib to explore and visualize the basic attributes of the data, such as size, sparsity, cleaniness and distribution of the values of categorical columns. The purpose is to ensure the data that will be later used for training is in a good state. For example, I want to make sure the number of categories is reasonable to avoid the curse of dimensionality. (if tail is too long I can lump tail values into a single value)

Second, I will use sklearn's one-hot encoder to encode categorical features. I will use tf-idf to vectorize description field. I will utilize Tf-idf's optional features for text normalization such as converting text to lowercase and to remove english stop words. After the vectors for different features are generated, I will concatenate them together into one single feature vector. The labels are simply the numeric score of each wine. I will split the features and labels into training and testing sets.

Next, I will use the training set to train an SGD regressor using sklearn. For benchmark, I will train a linear regression model using sklearn on the same training set.

Finally, I will use the trained SGD regressor and linear regressor to predict the scores of the testing data separately, and feed the predicted scores and the actual scores to sklearn's r2 score module to calculate the r2 scores of both methods as a measure of success.

References

[1] Wine production in the United States and in California from 2006 to 2016 (in million gallons)*, `https://www.statista.com/statistics/259493/wine-production-in-the-us-and-california/`

[2] Mordor intelligence: Global Wine Market - Growth, Trends and Forecasts (2017 - 2022), `https://www.mordorintelligence.com/industry-reports/wine-market`

[3] Wine Business Monthly - Number of United States Wineries Reaches 9,091, `https://www.winebusiness.com/wbm/?go=getArticleSignIn&dataId=179530`