Wine Variety Classification by Wine Description with Multinomial Naive Bayes And Logistic Regression

Final Project Report | COMP 562 Fall 2019

VIJAY RACHAKONDA, NICK COLVIN, MARY HALVORSEN

ACM Reference Format:
Vijay Rachakonda, Nick Colvin, Mary Halvorsen. 2019. Wine Variety Classification by Wine Description with Multinomial Naive Bayes And Logistic Regression : Final Project Report | COMP 562 Fall 2019. 1, 1 (December 2019), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This report will discuss the prediction of wine variety and province based on wine descriptions provided by a dataset of wine reviews. This project was formed by first finding a dataset that provided text data that would allow classification based on another parameter. Inspiration was drawn from the third class assignment where the task was to attribute authors to articles by clustering articles and then perform statistical tests on each article cluster. The wine review dataset was selected because of the collection of wine descriptions that contained common words used in wine descriptions, which would allow a correlation to be drawn from significant word use across description with wine variety. Similar to our previous class assignment, two models were , the Multinomial Naive Bayes Model and a Logistic Regression Model. We chose Multinomial Naive Bayes because there are many possible varieties of wine within the data that each description could be classified as. The Multinomial Naive Bayes model also provided a unique level of analysis of the data set in that we could assume that given a particular variety, the features are independent. We based our observations on the accuracy scores of each model, as well as data taken from classification reports generated for each model.

2 SURVEY OF RELATED WORK AND SOURCES

Once the wine reviews data set was selected using Kaggle (1), we came across a model created by user Koji that had been built using natural language processing and the kMeans algorithm to attribute wine variety based on wine description (2). After removing the stop words and miscellaneous punctuation from the descriptions, Koji used the term frequency inverse document frequency (tf-idf) provided by sklearn to vectorize the token words used in the descriptions. Koji's findings were that the kMeans did not result in a one-to-one relationship with wine variety and description, but still produced meaningful clusters that gave a good representation of what a typical description for a certain wine variety could be. Koji proposed that another possible model to use for the wine varieties dataset would be the Multinomial Naive Bayes model, which provided some of our inspiration to use that in our own implementation. For the implementation of the Naive Bayes model and initial approach, the steps outlined in a video tutorial were used (3). This tutorial provided an outline of steps to follow for our chosen implementation: Import the data, vectorize the word counts, perform term frequency and inverse document frequency on the data, run the Multinomial Naive Bayes classification model. This tutorial also provided an in-depth explanation for each step.

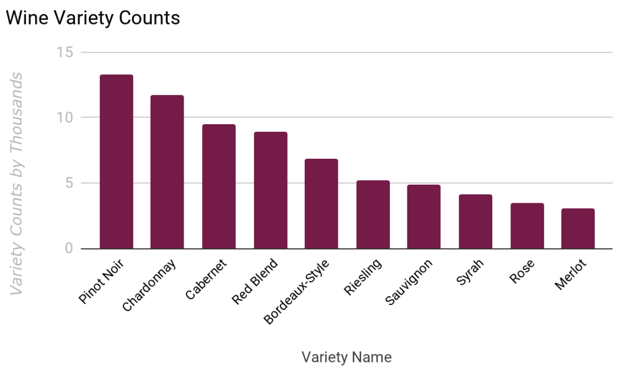


Fig. 1. Display of Counts for Wine Variety

3 METHODS AND APPROACH

3.1 Multinomial Naive Bayes

Our initial motivation for trying the Multinomial Naive Bayes classifier was again after reading through Koji's approach for NLP using the K-means algorithm for clustering related word groupings together. He then created a heat map to show how prevalent each wine variety was in the 15 word groups. He found that certain wines like Chardonnay, Pinot Noir, and Chardonnay were represented strongly in certain clusters but other clusters were harder to classify. Koji remarked in the conclusion of his Kernel that it would be interesting to see how well the Multinomial Naive Bayes classifier would predict wine varieties from just the review description.

With a little bit of research we found out that Multinomial Naive Bayes is a common method used in NLP because of the Naive assumption that the features, in this case word frequencies and TF-IDF, are independent of each other. Below is a set of equations that define the Multinomial Naive Bayes approach in our use case.

Author's address: Vijay Rachakonda, Nick Colvin, Mary Halvorsen.

$$P(Variety|Description) = \frac{P(Description|Variety)P(Variety)}{P(Description)}$$

This equation above is a very high-level abstraction of what is going on under the hood of the scikit-learn model. This is the simple Bayes' Theorem representation of the problem.

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

If one were to actually break down the term $P(Description|Variety)$, it would look like the equation above, courtesy of scikit-learn user guide (4). $\hat{\theta}_{yi}$ represents the maximum likelihood of a certain feature, in this case a certain word, appearing in a class i, which in this case is a wine variety. N_{yi} is the number of times a certain feature i appears in class y and N_y is the count of all the features in class y. The alphas in the equation are for the purposes of smoothing, that is when words are absent in certain varieties it prevents the entire probability of $P(Description|Variety)$ going to zero.

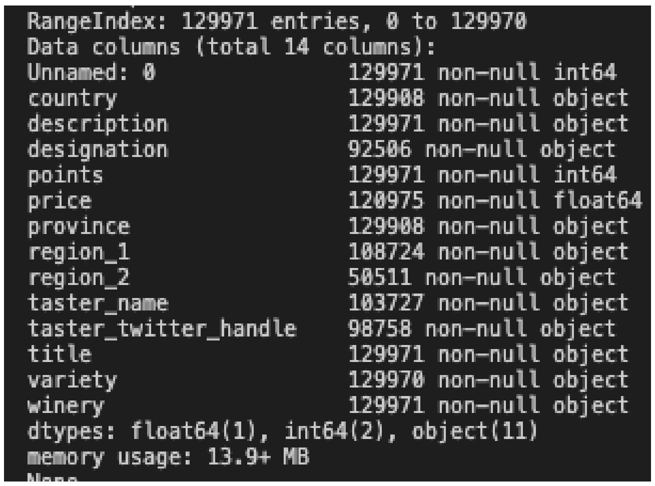
3.1.1 *Steps we took:*

(1) Data Preprocessing - Here we used Pandas to read in our csv file. Fig. 2 is a snapshot of how our data is laid out. Note here we only want the description and variety columns as description will contain our features and variety is being predicted. We also filtered down the number of wine varieties to the top ten for ease as there are over 1500 wine varieties in the file.

(2) Count Vectorization - Before being able to actually predict wine from wine descriptions, we needed to get the data into a format that supports that process. Similar to HW 3, we need to tokenize all of the words in our dataset. Essentially, that is assigning a unique numeric value to all of the words in our data set and instantiating an n long vector (n = number of words in the dataset) to store our words at each index (the index is representing our unique numeric values. The next task we had to do was being able to turn the tokenized array into a dictionary of tuples (row number, tokenized word numeric value) as the key and the counts in that particular row description as the value. This is accomplished with the `CountVectorizer.fit_transform()` method. Prior to doing either of those tasks, however, we must account for stop words, punctuation, common words in our dataset like wine or flavor, and any other extraneous symbols that do not aid in training our model. Note: We did use and modify Koji's punc array to construct our array of punctuation symbols, common words like wine and flavor, and extraneous symbols. We named this array `aux_stops`.

(3) TF-IDF Transformation - Now that we have the list, technically dictionary, of counts, we want to transform these counts into a more descriptive value, that is TF-IDF. TF-IDF is essentially applying less weight on the words that appear frequently across all documents. All we had to do here is use a TF-IDF transformer to simply convert the counts into their individual TF-IDF values. In summary, the keys of the dictionary stay the same but their counts are now TF-IDF values.

(4) Fitting Multinomial Naive Bayes model - Calculating the posterior probabilities of all of the wine varieties was handled by scikit-learn. We set our training-test split to 0.2, i.e. 80% of data was used in our training set and 20% was used to test our model. We did not have to add prior probabilities of each wine variety in the data set as scikit-learn calculates that automatically according to the data it reads from the training and test data sets. We did add a Laplace smoothing parameter of 0.1, so if we had any terms that never showed up in a given variety the $P(Description \mid Variety)$ would never go to zero.



```
RangeIndex: 129971 entries, 0 to 129970
Data columns (total 14 columns):
Unnamed: 0               129971 non-null int64
country                  129908 non-null object
description              129971 non-null object
designation              92506 non-null object
points                   129971 non-null int64
price                    120975 non-null float64
province                 129908 non-null object
region_1                 108724 non-null object
region_2                 50511 non-null object
taster_name              103727 non-null object
taster_twitter_handle    98758 non-null object
title                    129971 non-null object
variety                  129970 non-null object
winery                   129971 non-null object
dtypes: float64(1), int64(2), object(11)
memory usage: 13.9+ MB
```

Fig. 2. Description of the Wine database

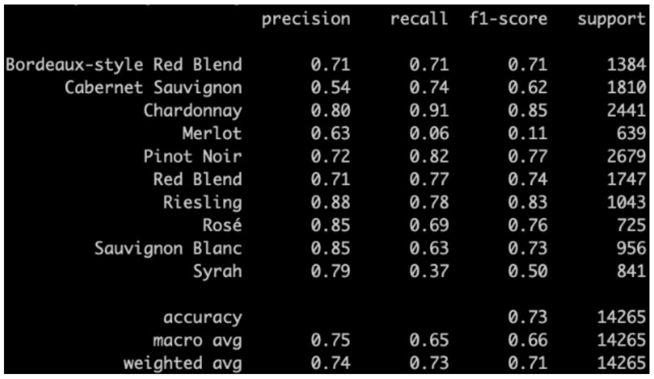
3.2 Logistic Regression

Given that this problem is a supervised classification one, we thought it might be a good idea to also implement a Multi-class Logistic Regression model in scikit-learn. A benefit in using the Multi-class LR model is that we do not have assume independence among features like we have to in Naive Bayes.

3.2.1 Steps we took: We took the same Data preprocessing steps for Logistic Regression as we did for the Multinomial Naive Bayes model. We fit it again with the dictionary of TF-IDF tuples and values. The only hyperparameter we added to the Logistic Regression model was L-2 regularization.

4 RESULTS AND CONCLUSIONS

The Multinomial Naive Bayes model produced an approximate 72% accuracy, and the Logistic Regression model produced an approximate 80% accuracy. We also ran classification reports using python sklearn on the predicted varieties for both sets of predicted wine varieties and for the varieties test data. The classification report allowed us to examine precision values for each wine variety. Specifically, the classification report provided analysis for the difference in accuracy between our two models. The classification report for the MNB model revealed large disparities between the F1-scores

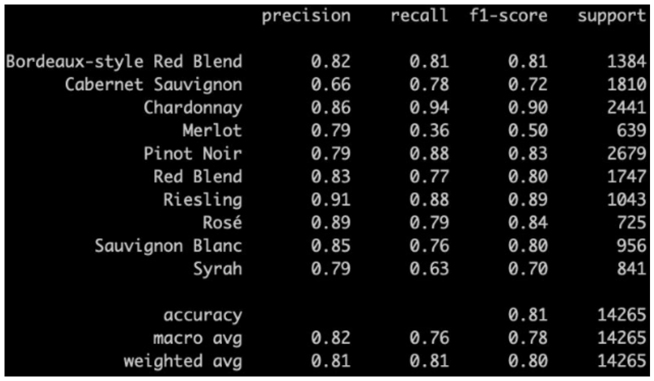


```
                          precision    recall  f1-score   support

Bordeaux-style Red Blend       0.71      0.71      0.71      1384
      Cabernet Sauvignon       0.54      0.74      0.62      1810
              Chardonnay       0.80      0.91      0.85      2441
                  Merlot       0.63      0.06      0.11       639
              Pinot Noir       0.72      0.82      0.77      2679
               Red Blend       0.71      0.77      0.74      1747
                Riesling       0.88      0.78      0.83      1043
                    Rosé       0.85      0.69      0.76       725
         Sauvignon Blanc       0.85      0.63      0.73       956
                   Syrah       0.79      0.37      0.50       841

                accuracy                           0.73     14265
               macro avg       0.75      0.65      0.66     14265
            weighted avg       0.74      0.73      0.71     14265
```

Fig. 3. Classification report for Multinomial Naive Bayes



```
                          precision    recall  f1-score   support

Bordeaux-style Red Blend       0.82      0.81      0.81      1384
      Cabernet Sauvignon       0.66      0.78      0.72      1810
              Chardonnay       0.86      0.94      0.90      2441
                  Merlot       0.79      0.36      0.50       639
              Pinot Noir       0.79      0.88      0.83      2679
               Red Blend       0.83      0.77      0.80      1747
                Riesling       0.91      0.88      0.89      1043
                    Rosé       0.89      0.79      0.84       725
         Sauvignon Blanc       0.85      0.76      0.80       956
                   Syrah       0.79      0.63      0.70       841

                accuracy                           0.81     14265
               macro avg       0.82      0.76      0.78     14265
            weighted avg       0.81      0.81      0.80     14265
```

Fig. 4. Classification report for Logistic Regression

of some variety classifications. The F1-scores in the classification reports reflect both the precision and recall values for each classification of wine variety and provide a meaningful analysis of the data. Chardonnay resulted in an 85% F1 score, while the merlot and syrah varieties produced F1 scores of 50% and 15% respectively.

The classification reports show similarities between the models. Both models shared varieties with the lowest two precision and recall scores: merlot, cabernet and merlot, syrah respectively. Furthermore, the MNB model recall values for each variety directly correspond with their respective frequency except for syrah and pinot noir. However, it is worth exploring why these models have the worst performance in the same two varieties.

The confusion matrices for both models along with the classification reports show merlot and cabernet were the least precise varieties of wine because they most often predicted each other incorrectly. The MNB model incorrectly predicted cabernet for 340 out of 639 merlot samples and predicted only 40 correctly, while logistic regression incorrectly predicted cabernet for 181 merlot samples and 269 correctly. Both models also most often incorrectly predicted the same variety, sauvignon blanc, instead of syrah, and MNB made this incorrect prediction more often than predicting correctly.

Improving the accuracy of the MNB model would be the next best possible step, starting with better balancing of weights for each class. Logistic regression performed much better but could also be improved with some form of regularization, although L-2 regularization did not make much of a difference when implemented.

REFERENCES

1. Thoutt, Zack. [zackthoutt]. (2017). Wine Reviews. Retrieved from https://www.kaggle.com/zynicide/wine-reviews/kernels

2. [Koji]. (2017). Exploring Wine Descriptions with NLP and kMeans. Retrieved from https://www.kaggle.com/kitakoj18/exploring-wine-descriptions-with-nlp-and-kmeans

3. [CodeWrestling]. (2018, October 21). Naive Bayes in Machine Learning Program | Text Classification python (2018) [Video File]. Retrieved from https://www.youtube.com/watch?v=0kPRaYSgblM&t=712s

4. [Scikit-learn] (n.d.). 1.9. Naive Bayes. Retrieved December 9, 2019, from https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes.