Predicting Wine Varieties from Text Descriptions

Yuan Wang
Princeton University
yuanwang@princeton.edu

Abstract

The blind-tasting portion of the Master Sommelier's exam requires a candidate to use "deductive tasting" to infer a wine's vintage, varietal, and origin within 25 minutes. Such process requires highly-trained skills and is laborious. We are interesting in whether we can model the deductive tasting technique in a data-driven way. Although computers cannot taste the wine, we train models that take the input of a corpus of wine descriptions and predict the label of the corresponding grape variety. We use the dataset scraped by Zack Thoutt and train four types of multi-class classifiers: multinomial naive Bayes (baseline), logistic regression, extreme gradient boosting (XGBoost), and bidirectional long short-term memory (LSTM) neural network. We explored different word representations including TF-IDF and GloVe. We selected top 20 most representative varieties for prediction and evaluated model performance with accuracy, F1 score, log-loss, area under the ROC curve, and area under the Precision-Recall curve. Overall, we found that logistic regression and XGBoost with TF-IDF representation produce competing performances, among which logistic regression has slightly higher accuracy and F1 score, and lower log-loss with TF-IDF (we name the model LG/TF-IDF). We further inspected the LG/TF-IDF model and discussed important features with domain-specific knowledge.

1 Introduction

In today's wine service industry, a Master Sommelier is the highest diploma that an individual can attain. These are professionals who are rare and share a proven mastery of the art, science, and history of the wine world. Since the first Master Sommelier examination in UK in 1969, by far there are only 255 professionals worldwide who have received the title of Master Sommelier [1].

One portion of the Master Sommelier exam is blind tasting, in which a candidate is confronted with six wines and has 25 minutes to identify the vintage, grape varietal, country, region and appellation using a very specific tasting technique called "deductive tasting [3]." To pass this exam requires extensive training and practice coupled with high financial costs. Therefore, in this project, we wonder if we can build machine learning models that, given documents of wine descriptions, can predict the varietal label of the corresponding wine in a data-driven rather than a laborious way. A grape variety is what people commonly used to refer to different wine categories, such as Pinot Noir or Chardonnay.

In recent years, gradient boosting and deep learning techniques are gaining unprecedented popularity in the machine learning community. In this project, we are interested to see how each technique performs in predicting wine varieties. We decided to test extreme gradient boosting (XGBoost), an implementation of boosting method, and long short-term memory (LSTM), an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Alongside of these two complex methods, we also tested multinomial naive Bayes and logistic regression, the former of which we use as the baseline model. Altogether, we have four classifiers for benchmarking.

The dataset we use was hosted on Kaggle by Zack Thoutt, who scraped nearly 150k wine reviews from WineEnthusiast during the week of June 15th, 2017 [2][4]. This dataset contains 10 variables

and the two variables of our interests are **Variety** and **Description**. Figure 1 gives a demonstration of the dataset. There is no missingness in the data. The raw data is a $150,930 \times 11$ matrix; after preprocessing (**Section 3**) and mapping the text descriptions to word embeddings (**Section 4**), we obtained a matrix of $71,200 \times 20,000$ for TF-IDF and another matrix of $71,200 \times 300$ for GloVe. We kept top 20 varieties and transformed them into 20 numerical labels $\{y = 0, 1, ... , 19\}$.

Therefore, our problem is: given an input matrix of size $71,200 \times 20,000$ or $71,200 \times 300$, we would like to predict, for each document, one of the 20 classes. The rest of the paper is organized as follows. **Section 2** introduced related works and motivations for our model. **Section 3** explains the dimensions and preprocessing for the dataset and performed exploratory analysis on the processed data. **Section 4** discuses the methods used, including the two word representation schemes and four classifiers. **Section 5** talks about the model performance and results. **Section 6** includes conclusion and discussion for future work. **Appendix** provides supplementary figures that facilitate the interpretations of previous sections.

country	description	designation	points	price	province	region_1	region_2	variety	winery
US	This tremendous 100% varietal wine hails from Oakville and was aged over three years in oak. Juicy red-cherry fruit and a compelling hint of caramel greet the palate, framed by elegant, fine tannins and a subtle minty tone in the background. Balanced and rewarding from start to finish, it has years ahead of it to develop further nuance. Enjoy 2022–2030.	Martha's Vineyard	96	235.0	California	Napa Valley	Napa	Cabernet Sauvignon	Heitz

Figure 1: Example of the wine dataset.

2 Related work

There is very little work that applied machine learning techniques to the domain of classifying or predicting wines. Using the same dataset, several bloggers performed exploratory analyses on wines and published their results on popular data science blogs. For example, Aleszu Bajak focused his analysis on French wines by using R to map each word in the vocabulary to a TF-IDF representation [7]. With those, he create barplots per each French province to surface the top unique keywords of wines in each region. He further created maps of France and marked the mean price of wines produced by each region with differently sized dots on the map. Anther data science blogger, John Bencina, published an article on exploring and classifying wine reviews [8]. Unlike Aleszu who only cared for French wines, John's interest was broader. He plotted the wine points distributions of top 20 regions in the dataset along with a heatmap that highlights the highest scoring wine varieties within each region. He concluded that "Napa Valley receives high marks for its Cabernet Sauvignon, Red Blend, Malbec, and a couple of white wine blends," and that the "Walla Walla Valley takes first across a few white varieties like Chardonnay, and Sauvignon Blanc and also a couple of reds like Tempranillo." To classify wines, he used a TF-IDF representation of descriptions as input and performed hierarchical clustering.

The most relevant work to our project is by Tim Aiken and Clara Meister, a pair of students at Stanford University who used Zack Thoutt's wine dataset for natural language processing [6]. Among all wine varieties, they chose the top 30 most represented varieties and labeled all else with "other". They then trained a bidirectional LSTM with 100 hidden units using a pre-trained 200D GloVe embedding to predict the 31 categories. They carefully evaluated several network architectures including gated recurrent units (GRUs). They also experimented by adding a convolutional layer followed by a max-pooling layer on top of the current network. However, they eventually chose the current network for both performance and simplicity. Because of that, we decided to adopt their network architecture in our implementation of LSTM.

3 Dataset

3.1 Preprocessing

To preprocess the data, we first removed unused columns and duplicated rows. This reduced about 50K rows and resulted in a $97,821 \times 2$ matrix. Since our dataset contained more than 600 wine varieties, to reduce classification complexity, we selected top 20 highly represented and most commonly known wine varieties. This yielded 20 classes in total, which are: { 0: Bordeaux-style Red Blend, 1: Cabernet Sauvignon, 2: Chardonnay, 3: Corvina, Rondinella, Molinara, 4: Malbec, 5:

Merlot, 6: Nebbiolo, 7: Pinot Noir, 8: Portuguese Red, 9: Red Blend, 10: Riesling, 11: Rose, 12: Sangiovese, 13: Sauvignon Blanc, 14: Shiraz, 15: Sparkling Blend, 16: Syrah, 17: Tempranillo, 18: White Blend, and 19: Zinfandel}. The final dataset is has dimension of 71,200 rows and 2 columns, namely "descriptions" and "variety". We split the data with stratified sampling into 80/20 train set and test set, such that the train set contains 56,960 documents and the test set includes 14,240 documents.

3.2 Exploratory analysis

Although we did not use points and price in our analysis, we would still be interested in obtaining an idea of the overall wine quality through these two variables. Table 1 shows some basic statistics of the points and price for all wines in the dataset.

	Mean	**Std**	**Min**	**25%**	**Median**	**75%**	**Max**
Points	88.05	3.30	80	86	88	90	100
Price($)	35.32	38.96	4	16	25	42	2,300

Table 1: Summary of points and price for all wines.

We found that there are 10 different wines that cost $4. Majorities were produced from a California winery called 'Bandit' with varieties including Cabernet Sauvignon, Chardonnay, and Merlot. The most expensive wine is a Bordeaux-style Red Blend from the famous French wine estate, Château Latour, and has earned 99 points from tasters.

We then selected four most widely consumed wines - two red and two white - and plotted the corresponding distributions of point and price (Figure 2). We can see that Pinot Noir and Cabernet Sauvignon are similar both in terms of quality and costs. The most expensive Chardonnay costs $2,013, which clearly skewed its price distribution. Sauvignon Blanc is comparatively cheaper and of worse quality.

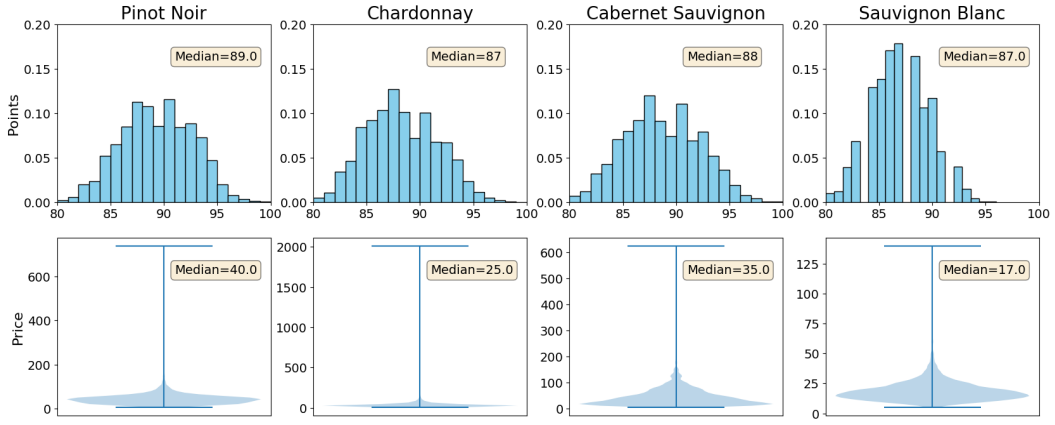

Figure 2: Exploratory analysis of points and price for selected wines.

4 Methods

4.1 Word representations

We explored two different word representations: term frequency-inverse document frequency (TF-IDF) and global vectors for word representation (GloVe) [14].

4.1.1 Term frequency-inverse document frequency (TF-IDF)

The TF-IDF has two components. The **term frequency (TF)** gives us the frequency of the word in each document in the corpus. Let i denote each word and let j denote each document for a total of N documents. The TF weight is calculated as

$$\text{tf}_{i,j} = \frac{n_{i,j}}{\sum_{j \in N} n_{i,j}}, \tag{1}$$

where $n_{i,j}$ is the number of times word i appears in document j. The **inverse document frequency (IDF)** measures how much weight a word carries. It is given as

$$\text{idf}(i) = \log(\frac{N}{df_i}), \tag{2}$$

where df_i is the total number of documents containing word i. Together, the TF-IDF weight of each word i in document j equals

$$\text{tfidf} = \text{tf}_{i,j} \times \text{idf}(i). \tag{3}$$

In our implementation, we used the `TfidfVectorizer` functionality of Python scikit-learn and retained top 20,000 words as our vocabulary [13].

4.1.2 Global vectors for word representation (GloVe)

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence matrix X from a corpus, whose entries $X_{i,j}$ tabulate the number of times word j occurs in the context of word i. In our case, we used the version trained on Wikipedia 2014 + Gigaword5 (6B tokens, 400K vocab, 300D vectors).

The resulting representations showcase interesting linear substructures of the word vector space [14]. The most general model takes the form,

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{i,k}}{P_{j,k}}, \tag{4}$$

where $w \in \mathbb{R}^d$ are word vectors, $\tilde{w} \in \mathbb{R}^d$ are separate context word vectors, and $P_{i,k}$ is the probability that word k appear in the context of word i. The authors further restricted the model based on three aims such that the final model adapted is,

$$F\big((w_i - w_j)^T \tilde{w}_k\big) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)}, \tag{5}$$

which exploits the inherent linear structure of vector spaces. The intuition behind the word-vector differences and the linear substructure is that, for example, we expect the differences between *man - woman, king - queen*, and *brother - sister*, i.e. words of opposite genders, to all be roughly equal (see **Appendix** Figure 6 for a graphical illustration) [14].

4.2 Classifiers

We tested four multiclass classifiers: multinomial naive Bayes (NB), logistic regression (LR), extreme gradient boosting (XGBoost or XGB) [10], and bidirectional long short-term memory (LSTM).

4.2.1 Multinomial Naive Bayes (NB)

We used the multinomial naive Bayes classifier because it produced a better result than Gaussian naive Bayes and took much less runtime. This algorithm is designed for discrete and multinomially distributed data, and is most commonly used in text classification [13]. The multinonimial distribution is parametrized by vectors $\theta_y = \{\theta_{y1}, ..., \theta_{yn}\}$ for each class y and vocabulary size n. A naive Bayes classifier assumes that features are conditionally independent given the class label, therefore the class conditional density is a product of one dimensional densities [12]:

$$p(\vec{w}|y = c, \theta) = \prod_{t=1}^{n} p(w_t|y = c, \vec{\theta_{tc}}). \tag{6}$$

In practice, we used the `MultinomialNB` functionality of scikit-learn with default parameters. Notice that by default, we have set the smoothing prior $\alpha = 1$, which is the Laplace smoothing. We only used the TF-IDF representation on NB classifier because it does not support negative input.

4.2.2 Logistic regression (LR)

Logistic regression is a discriminative model which is linear in the parameters. It uses a logistic function, which takes the form,

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{7}$$

to output the probabilities of the possible outcomes of a single trial. We implemented the `LogisticRegression` class of scikit-learn by specifying the solver to be 'lbfgs' coupled with an ℓ_2-regularization that prefers sparse solutions, and setting the `multi_class` option to 'multinomial'. According to scikit-learn, the 'lbfgs' solver is found to converge faster for some high-dimensional data. Moreover, setting `multi_class` to multinomial with this solver uses the cross-entropy loss and learns a true multinomial logistic regression model [9]. We further set `random_state` to 0 and `max_iter` to 100, and kept all other parameters as default. We tested both TF-IDF and GloVe representations on LR.

4.2.3 Extreme gradient boosting (XGBoot or XGB)

XGBoost is a scalable end to-end tree boosting system which has been widely used in and champions a wide range of online data science challenges and competitions. Gradient tree boosting algorithms are a family of highly effective machine learning method. Given a dataset with n observations and m features, $\mathcal{D} = \{(\mathbf{x}_i, y_i\}(|\mathcal{D}| = n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, a tree ensemble model uses K additive functions to predict the output,

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), f_k \in \mathcal{F}, \tag{8}$$

where $\mathcal{F}$ is the space of regression trees known as CART [10]. To learn the set of functions, the model is trained in an additive manner and minimizes some regularized learning objective. In our implementation, we used the `xgboost` library of Python which was first developed by Chen, et al. To tune our hyperparameters, we ran a small-scale 5-fold cross validation from the parameter set {learning_rate = {0.1, 0.3}, n_estimators = {100, 200, 300} }. We were unable to fine-tune our model due to computational constraints. After grid search, we set `learning_rate` to 0.3 and `n_estimators` to 200. We also set early-stopping of 10 rounds to prevent the model from overfitting. The learning objective is 'multi:softprob' while all else remained default. We tested both TF-IDF and GloVe representations on XGBoost.

4.3 Bidirectional long short-term memory (LSTM)

LSTM is a recurrent neural network (RNN) architecture. It is appropriate for sequential data, such as text descriptions in our case. According to previous literatures, bidirectionality is useful because later words may influence the weight of an input word. Figure 3 shows the architecture of our model, which is borrowed from Aiken et al. [6]. The Dropout layer is used to prevent over-fitting as well as reducing training time. The final Dense layer of hidden size 20 is used to classify an example into one of the 20 labels. We trained the model using an Adam optimizer and the batch size was default to 32. We also used categorical cross-entropy loss, which has been widely suggested for multi-class classification tasks. Mathematically, the cross entropy function computes

$$H(p, q) = -\sum_{x} p(x)\log(q(x)), \tag{9}$$

where p is the true probability distribution and q is an estimated probability distribution. The ReLU activation function, $y = \max(0, x)$, is the most commonly used activation function in artificial neural networks. It is cheap and converges fast, and the lower-bound of 0 promotes sparse activation. The softmax activation function maps an input of real-valued vector to a probability distribution. A standard softmax function, σ, takes the form,

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}}. \tag{10}$$

We implemented the LSTM model with Keras [11]. During training, we set the number of epochs to 30 and used the test set as validation data. We employed early-stopping by monitoring validation loss and setting `patience` to 3. Aiken et al. mentioned that previous literatures suggested that TF-IDF is not appropriate for LSTM, therefore, we only tested the GloVe embedding on this model.

4.4 Metrics

As a multi-class classification problem, we used accuracy, weighted F-1 score, area under ROC curve for each class, area under Precision-Recall curve for each class, and log-loss as the evaluation metrics. To establish the definitions of each of these terms, we first denote the number of true

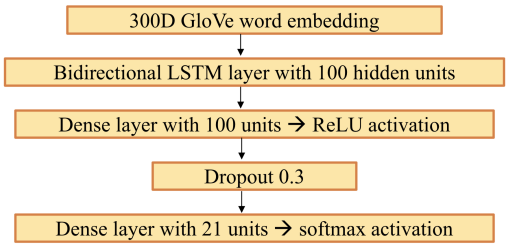

Figure 3: Architecture of the bidirectional LSTM model.

positives, true negatives, false positives, and false negatives as TP, TN, FP, and FN, respectively. Therefore, we have

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \text{ recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \text{ and accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}. \tag{11}$$

We can generate the ROC curve by plotting the true positive rate (TPR) against the false positive rate (FPR) such that a greater area under the ROC curve (auROC) is preferred. Similarly, a greater area under the Precision-Recall curve (auPRC) is preferred. TPR, FPR, and F1 score are each defined as

$$\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \text{ FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}}, \text{ and F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{12}$$

Log-loss is defined on probability estimates [13]. It is the same as cross-entropy loss. Below we use the terminology defined in scikit-learn to formulate the mathematical expression. Let the true labels be encoded as a binary indicator matrix Y, and let P be a matrix of probability estimates with $o_{i,k} = P(t_{i,k} = 1)$. Then the log loss in this multi-class setting is,

$$L_{\log}(Y, P) = -\log P(Y|P) = -\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k} \log p_{i,k}. \tag{13}$$

5 Results

5.1 Model evaluations

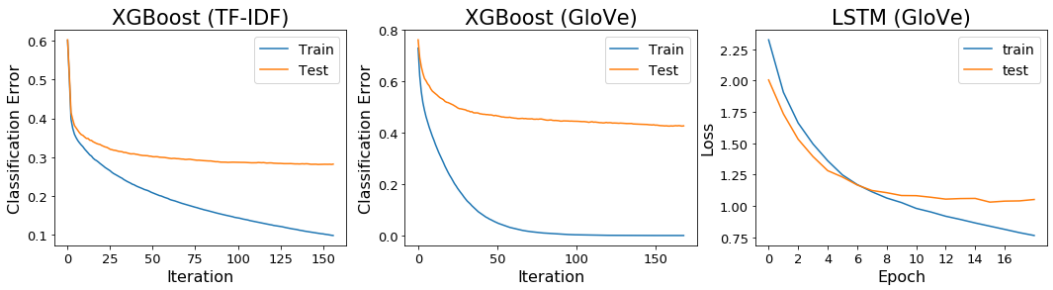

Figure 4: Convergence of XGBoost and LSTM. XGBoost measures `merror`, which is calculated as $N_{\text{wrong}}/N_{\text{all}}$. LSTM used categorical cross-entropy loss.

We are interested in evaluating the convergence of XGBoost and LSTM models because they are more complicated and harder to train than the two shallow methods, NB and LR. Before, we set early-stopping for both models to prevent them from over fitting. After fitting XGBoost and LSTM, we found that (i) XGBoost with TF-IDF embedding stopped at the 157th iteration, (ii) XGBoost with GloVe embedding stopped at the 169th iteration, and (iii) LSTM with GloVe stopped at the 19th epoch. Figure 4 shows the convergence of training and testing error for each of the three models.

5.2 Prediction results

We first compare the accuracy (Acc), weighted F1 score (F1), and log-loss (LL) between each model we ran (Table 2). With the 20,000-word TF-IDF representation, we found that logistic regression yielded the best performance, while XGBoost was only marginally worse than logistic regression. However, we would still prefer logistic regression given its simplicity and fast training time.

With the 300D GloVe embedding, bidirectional LSTM clearly outperformed LG and XGBoost. This was expected because we hypothesized that deep neural networks are able pick up the intrinsically complex structure in the GloVe embedding which shallow learning methods cannot.

Classifier	TF-IDF			GloVe		
	Acc	F1(w)	LL	Acc	F1(w)	LL
NB	0.51	0.42	1.64	-	-	-
LG	**0.73**	**0.72**	**0.94**	0.61	0.60	1.25
XGB	0.72	0.65	1.08	0.57	0.56	1.43
LSTM	-	-	-	**0.68**	**0.68**	**1.05**

Table 2: Evaluation results of four classifiers on two embeddings using various metrics. The F-1 score is weighted by support. A higher Acc or F1 is preferred, while a lower LL is better.

Since logistic regression with TF-IDF (LG/TF-IDF) yielded the most desirable result, we further inspected the performance of this model in terms of area under the ROC curve (auROC) and the area under the Precision-Recall curve (auPRC) (see Figure 7 in **Appendix A: Supplementary Figures** due to space constraint). We are able to achieve a mean auROC of 0.98 and a mean auPRC of 0.81. If using auAUC as the evaluating metric, the best-performing class (auROC=1.0) is 'Corvina, Rondinella, Molinara', which is the smallest class with 1,118 associated documents. The two worst classes are 'Merlot' and 'Red Blend' which are medium-sized, and that each has an auROC of 0.94 and 0.93. Overall, the model achieved satisfying results.

On the other hand, if using auPRC as the evaluating metric, the best-performing class (auPRC=0.94) is 'Chardonnay' which is the second largest class with 9,159 sample. The worst, 'Tempranillo', however, has only an auPRC of 0.57. Notice that it has a small class size of 1,622. Meanwhile, the smallest class 'Corvina, Rondinella, Molinara' achieved an auPRC of 0.89, showing that auPRC is not necessarily correlated with class size and that our imbalanced dataset did not cause significantly bias in our model. Nevertheless, with this metric, the inter-class performance varies more than that with auROC.

We then plotted the distribution of the predicted labels against the true labels along with a confusion matrix for the LR/TF-IDF model. From Figure 6, we can see that we have an imbalanced dataset with some over-represented classes, namely 'Pinot Noir', 'Chardonnay', and 'Cabernet Sauvignon'. However, our model is not significantly biased towards any one of these (the differences between percent predicted and percent true are less then 2%), indicating once again that that it did not default to big classes to boost accuracy.

Last, we were interested to see what features are important in classifying each variety. Here we demonstrate a subset of important words extracted from the LG/TF-IDF model in Table 3. We were surprised to see that our model is able to effectively capture some words that are distinctive for a particular variety, which are highlighted in italics. For example, the Chablis region of France produces Chardonnay grape; however, due to the cool climate of the region, the Chablis wine has established its fame through producing wines that have more acidity and less fruitiness than normal Chardonnay wines. In another example, Pommard is a commune in Burgundy, France. The Appellation d'origine contrôlée (AOC) Pommard is one of the most highly regarded red wine which uses Pinot Noir as the main grape variety [5]. Similar stories can be told for *Hermitage* and *Rhône* for the Syrah variety.

More interestingly, we saw that 'alcohol' was one of the most important features in predicting Zinfandel. This truly captures a unique feature of Zinfandel because this kind of wine contains higher level of alcohol than regular kinds. During the fermentation process of wine making, yeasts breaks down sugar in the grapes and produces alcohol and other biochemical compounds. Because of the high sugar content of the Zinfandel grape, it can be fermented into levels of alcohol exceeding 15 percent [15].

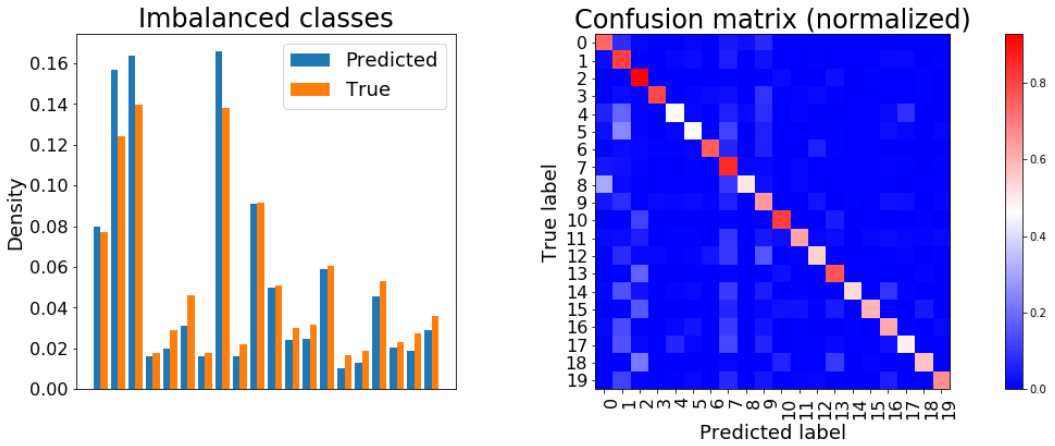

Figure 5: (a) Histogram of distribution of true and predicted labels using logisitic regression with TF-IDF embedding. (b) Confusion matrix of the same model.

Overall, we can conclude that our model is capable of finding the most relevant features for the classification task in this project.

Selected Variety	Important Features
Cabernet Sauvignon	tannis, blackberry, oak
Chardonnay	apple, *Chablis*, tropical
Merlot	cherry, plum, herb
Pinot Noir	silky, *Pommard*, earthy
Riesling	honey, citrus, minerality
Sauvignon Blanc	grapefruit, passion, lemongrass
Syrah	pepper, *Hermitage*, *Rhône*
Zinfandel	*alcohol*, blackberry, raspberries

Table 3: Summary of the important features in classifying selected varieties.

6 Conclusion and Discussion

In this project, we explored four different classifiers, from baseline models to deep neural networks, that take an input of a word-document embedding and predict the grape variety for each document of text descriptions of a wine. We transformed the original corpus of 71,200 documents into two word representations, the TF-IDF representation and the GloVe embedding. We used TF-IDF on multinomial naive Bayes (NB), logistic regression (LG), and XGBoost; and GloVe on LG, XGBoost, and bidirectional LSTM. We split the data into 80% train and 20% test and evaluated the performance of each model with metrics such as accuracy, F1 score, log-loss, auROC and auPRC. We found that both XGBoost and LSTM converged fast with early stopping. LG/TF-IDF yielded a globally highest performance with an accuracy of 0.73 and F1 of 0.72. Moreover, within each embedding scheme, LG outperforms the rest two with TF-IDF and LSTM outperforms the rest with GloVe. We further inspected the LG/TF-IDF model and explained the important features for each class in the context of background wine knowledge.

For future work, we hope to create a better labeling scheme instead of naively using top-20 most represented classes. By ground truth, some of the varieties can be re-labeled and incorporated into other highly-represented varieties, and we hypothesize that doing so will create more accurate labels thus further improve the classification results. We can also test different methods that overcome imbalanced dataset, including down-sample the over-represented classes or up-sample under-represented classes. We would also hope to fine-tune our XGBoost models if given more computing resource.

A Supplementary Figures

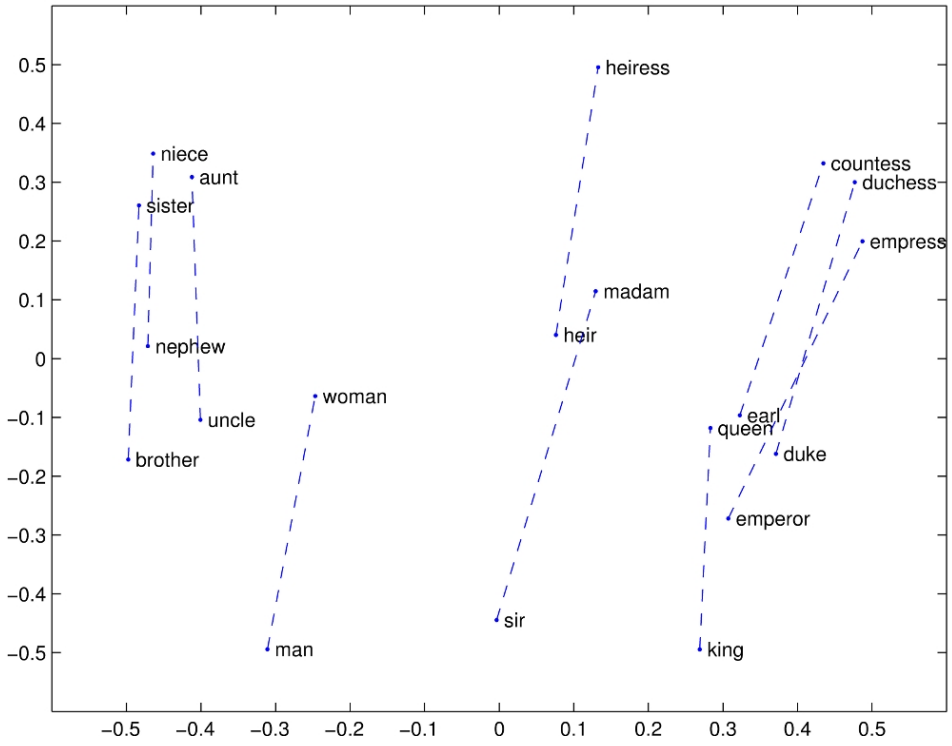

Figure 6: Example of the linear substructure of the GloVe embedding.

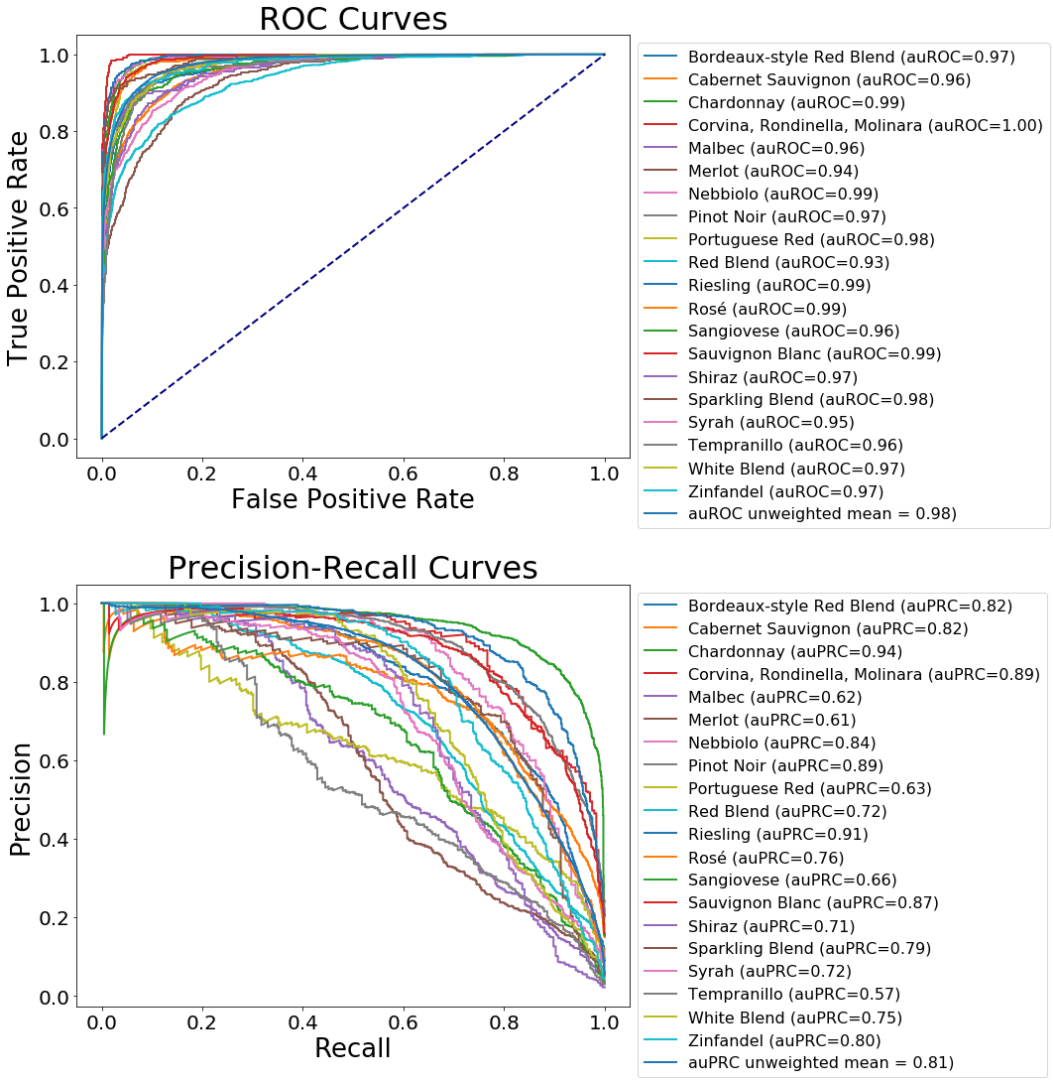

Figure 7: Logistic regression with TF-IDF embeddings. We included curves for all 20 classes plus the unweighted mean. Detailed explanations can be found in **Section 5.2** of the main write-up.

References

[1] Court of Master Sommeliers of America. https://www.mastersommeliers.org/about. Accessed: 2019-04-22.

[2] Kaggle: Wine reviews. https://www.kaggle.com/zynicide/wine-reviews. Accessed: 2019-04-22.

[3] Tim Gaiser, Master Sommelier. http://www.timgaiser.com/how-to-taste-wine.html. Accessed: 2019-04-22.

[4] Wine Enthusiast. https://www.winemag.com/?s=&drink_type=wine. Accessed: 2019-04-22.

[5] Pommard wine. https://www.mastersommeliers.org/about, Nov 2015. Accessed: 2019-05-13.

[6] Tim Aiken and Clara Meister. Applying Natural Language Processing to the World of Wine. *Stanford University, CS230*, 2018.

[7] Aleszu Bajak. Using French wine reviews to understand TF-IDF, a measure of how unique a word is to a document, Jun 2018.

[8] John Bencina and John Bencina. Exploring and Classifying Wine Enthusiast Reviews, Jul 2017.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[11] François Chollet et al. Keras. https://keras.io, 2015.

[12] K Murphy. Machine Learning: A Probabilistic Approach. *MIT Press*.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[15] Charles L. Sullivan. *Zinfandel: a history of a grape and its wine*. University of California Press, 2003.