Wine Type Prediction Using Sentiment Analysis and Price

Xiqiang Lin
A92118769
xil307@ucsd.edu

Xinrui Zhou
A92054932
x9zhou@ucsd.edu

Jinwei Ren
A92061199
jir017@ucsd.edu

0 Abstract

In this paper, we try to build a predictive model of determining the variety of a wine. The model behaves like a wine master to identify wine varieties through blind tastings. In order to train our model, we need to figure out information that is highly related with wine varieties. Wine description/reviews reflect tasters' opinions toward the wine. Thus, text mining is an efficient tool to find patterns in wine reviews of different varieties. Different sentiment tasks are implemented, including using unigram and tf-idf for SVM with different kernels. It turns out that SVM and sentiment analysis with most common unigrams gives higher accuracy compared to other models. In addition to reviews, we also try to add some numerical data, such as "price" to our feature vector. The model (our win master) will not be able to directly taste the wine but can identify the wine based on its description/review.

1 Introduction Dataset & Exploratory Analysis

The dataset is https://www.kaggle.com/zynicide/wine-reviews scraped from WineEnthusiast by a kaggle user, ID "zackthoutt". The dataset contains 130k samples data with 13 fields including *Points*: the number of points WineEnthusiast rated the wine on a scale of 1-100 (though they say they only post reviews for wines that score >=80);

Title: the title of the wine review, which often contains the vintage if you're interested in extracting that feature;

Variety: the type of grapes used to make the wine (ie Pinot Noir);

Description: a few sentences from a sommelier describing the wine's taste, smell, look, feel, etc.;

Country: the country that the wine is from;

Province: the province or state that the wine is from;

Region 1: the wine growing area in a province or state (ie Napa);

Region 2: sometimes there are more specific regions specified within a wine growing area (ie Rutherford inside the Napa Valley), but this value can sometimes be blank;

Winery: the winery that made the wine;

Designation: the vineyard within the winery where the grapes that made the wine are from; ***Price***: the cost for a bottle of the wine; *Taster Name*: name of the person who tasted and reviewed the wine; *Taster Twitter Handle*: Twitter handle for the person who tasted and reviewed the wine.

In this paper, because each wine is categorized by the type of it uses, we will be using "variety" and "category" interchangeably,.

Before processing data, we need to filter out some noises in order to get accurate results. There are three main issues of this dataset. First is that there are many duplicates; second is that many price fields are null; third is that the dataset is highly unbalanced: 85% wine varieties has < 100 reviews and 57% wine varieties has < 10 reviews. To resolve these issues, we decided to drop all duplicates based on the description column alone and also drop samples with missing price data. Some wine varieties do not contain enough data in the dataset for us to generate an accurate model for its prediction. Therefore, we decided to build a category prediction model that only predict top 10 varieties. After dropping observations that are not in these 10 varieties, there are 62150 samples data in our dataset. Data distribution over varieties is shown in Figure 1.1

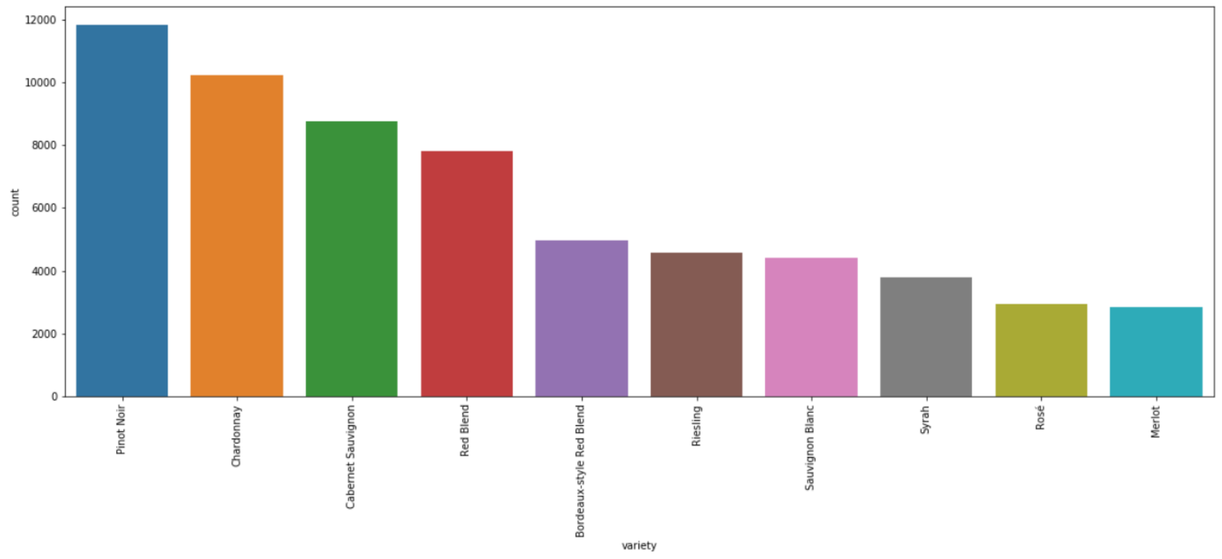


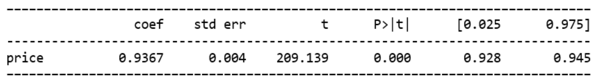
Figure 1.1 : Data distribution over varieties

The Distribution of Samples in Top 10 Varieties

In order to find which fields are most relevant to wine variety, we did some exploratory analysis. We already know that text mining of the description fields will be efficient to build our feature vectors. For each wine variety, most descriptions will contain same words to describe its aroma, taste, color and so on. Besides text, we also want to know how numerical data (price & points) is related with the wine variety.

Price VS Points

The Figure 1.2 indicates that there is a positive correlation between price of a wine and its points (ratings). If the rating of a wine increases by 1, its prices will on average increase by $0.94.



```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
price          0.9367      0.004    209.139      0.000       0.928       0.945
==============================================================================
```

Figure 1.2: Average price

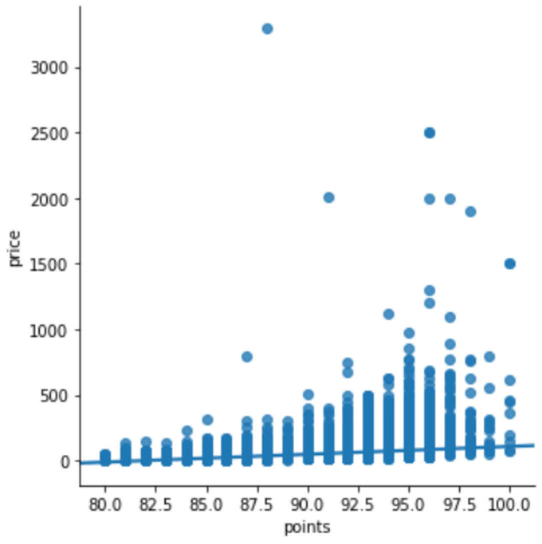


Figure 1.3: Correlation between price and points

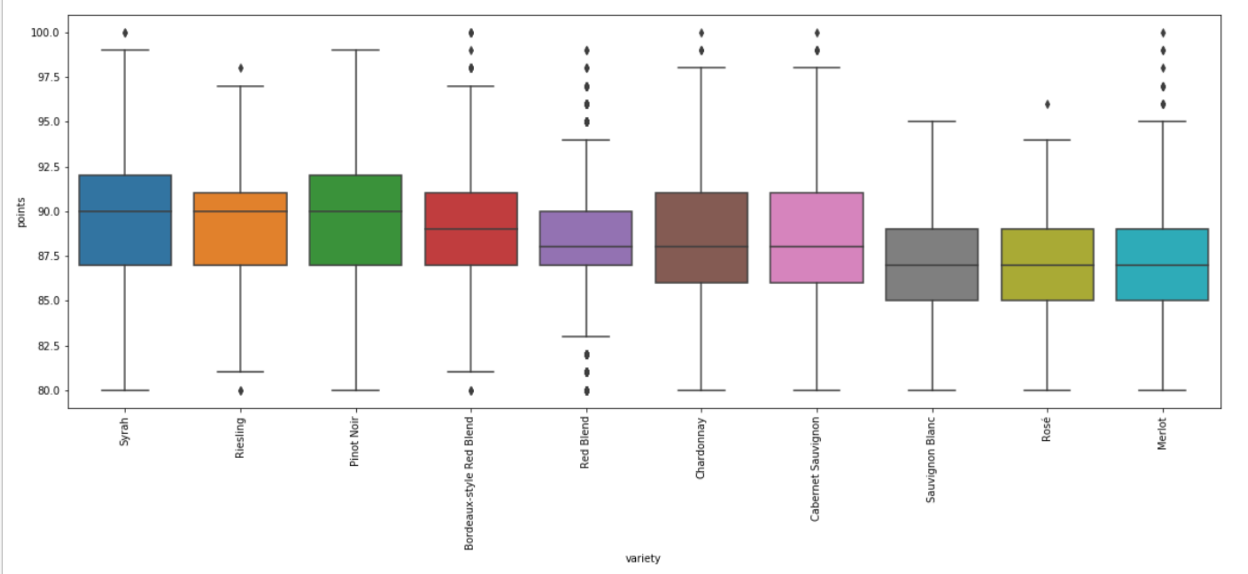


Figure 1.4: Points (rating) variation across for each variety

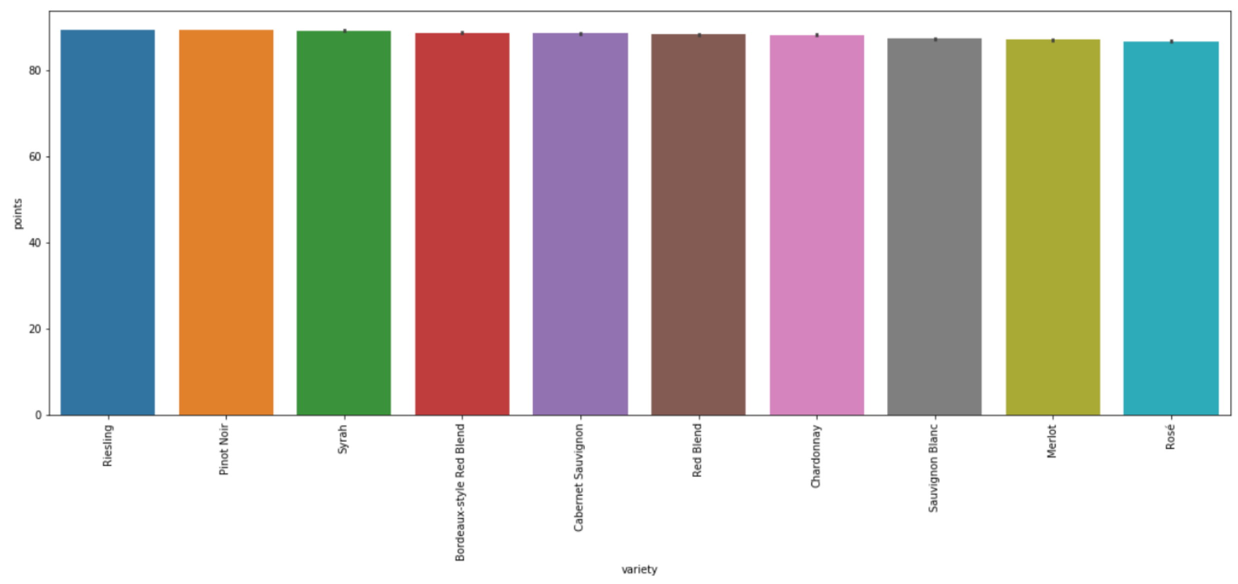


Figure 1.5: Mean point (rating) for each variety.

Points VS Variety

Figure 1.4 is a box-plot chart including top 10 wine varieties and their points distributions. Syrah appears to have the highest median points among all wine varieties. Red Blend and Merlot tend to have a large number of highly reviewed outliers. And Figure 1.5 shows mean rating points for each wine variety. Before we thought that points could be useful feature to reflect overall quality of each wine variety. However, since the distribution overall appears to be fairly uniform, we decide to not use 'points' to build our future models

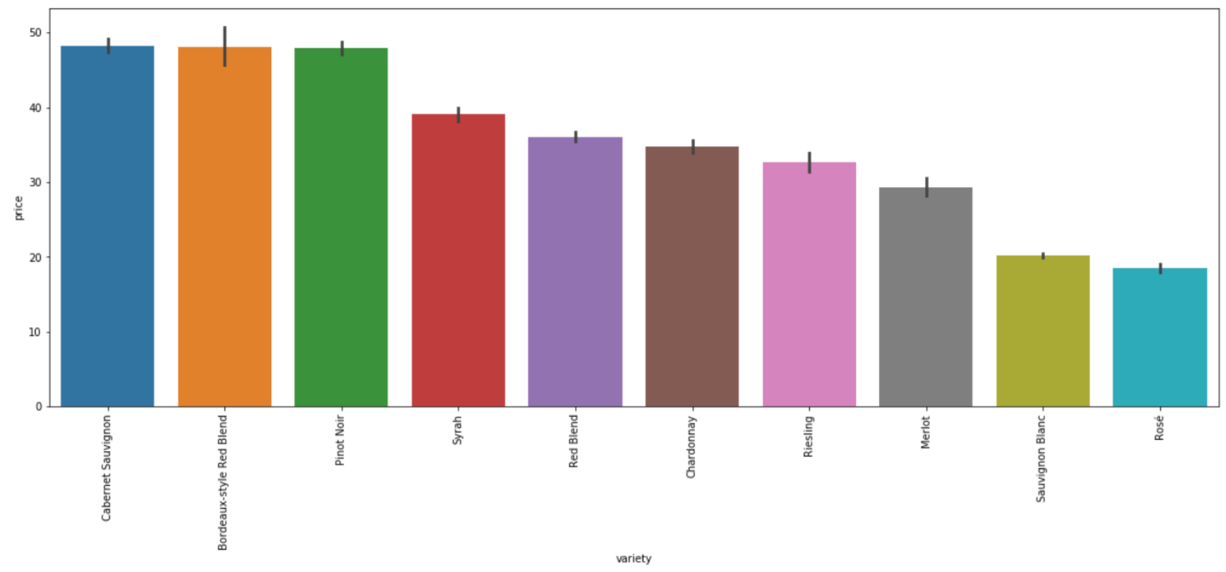


Figure 1.6: Mean price for each variety

Price VS Variety

Figure 1.6 shows the mean price of each wine variety, which confirms our assumption that the price can effectively help differentiate each wine variety. Because of the clear variation, we will try to include it in our future models and see how it will affect accuracy.

2.0 Predictive Task

Our predictive task is, given the data of a wine, we want to predict the variety of the wine based on other information of the wine. For example, for a data piece in which the "variety" is "Red Blend", we want to predict the wine's variety as 'Red Blend' in this case. To complete the predictive task, we use several models like:

1. Maximum Cosine Similarity
2. Logistic Regression
3. Linear Support Vector Machine

To evaluate those models, we design 2 baseline models to accomplish the task.

2.1 Baseline description

2.11 Baseline 1

We count the number of occurrence of each variety in the training set. Then we get the most frequent variety. Then, we predict the most popular variety regardless of any characteristic of data.

For example, the most popular variety of the wines is "Pinot Noir" in our training set. So, we return this variety directly each time when we make a prediction.

To evaluate this baseline, we apply the model to the test set, count how many wines actually belongs to the variety "Pinot Noir", and divide the count by the number of testing data.

The accuracy of this baseline is **0.187**.This model heavily depends on the ratio of the most popular variety, which means it is not very stable.

2.12 Baseline 2

We select keywords from the name of the variety for each of the 10 variety.

Then, we predict the category based on the keywords. For example, we predict “Pinot Noir” when the description contains “pinot” or “noir”, and we predict “Red Blend” when the description contains “red” or “blend”. If none of the keyword is found, predict the most frequent variety.

To evaluate this baseline, we apply this model to the testing set, count how many varieties are predicted right, and divide the number by the total number of testing data.

The final accuracy is **0.3345671742239421**. This is significant improvement from baseline 1. This result shows that the name of each category itself has fairly good predicting power.

Then, we develop some advanced models which can easily beat those baseline models, and we found that the ***Bag of Words + Linear Support Vector Machine*** model give the best accuracy.

3.0 Model Selection and Description

We proposed three different model for our categorization task. One based on cosine similarity, second one based on logistic regression, last one based on SVM. The reason why we chose a model based on cosine similarity is that when we model each document in terms of words they contain, documents of the same category is likely to have high similarity. The rationale behind choosing logistic regression is that this is a frequently used model by others who study this wine reviews dataset. It seems that logistic regression model has a high performance on predictive accuracy. For our own model, we would like to build on the success of logistic regression, linear model. So we choose SVM, a linear model that strives to minimize the number of misclassified points. The following section will explain the performance, strengths and weakness of each model.

3.1 Model: Cosine Similarity

The reason why we chose a model based on cosine similarity is that when we model each document in terms of words they contain, documents of the same category is likely to have high similarity.

Instead of focusing on the most common words first, we find which words are more common in each variety compared to others. First, compute the frequency of those 1000 common words as follows:

$$frequency_{word} = \frac{number\ of\ times\ the\ word\ appears}{\sum_{i=1}^{1000} number\ of\ times\ the\ ith\ most\ popular\ word\ appears}$$

Next, compute the above frequency per variety, e.g.:

$$frequency['Merlot']_{word} = \frac{number\ of\ times\ the\ word\ appears\ in\ 'Merlot'\ reviews}{\sum_{i=1}^{1000} number\ of\ times\ the\ ith\ most\ popular\ word\ appears\ in\ 'Merlot'reviews}$$

Having computed above, we can find which words are more popular in one variety compared to their overall frequency across all varieties, i.e.,

$$frequency['Merlot']_{word} - frequency_{word}$$

For each of the 10 varieties, we can get 10 words that are more frequent in that variety compared to other varieties. Then we will build feature vectors out of this 10 words * 10 varieties = 100 dimensions. First we build feature vector for each variety: ten words that are more frequent in that variety will have ‘1’ in vector and others will be ‘0’. E.g.:

‘Pinot Noir’: [1,1,1,1,1,1,1,1,1,1,0,0,0,0,..,0]

‘Chardonnay’:

[0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,..,0]

Thus, each variety has its own feature vector which represent the most frequent words in that variety. For each new review, we build the same 100 dimensional feature vector - ‘1’ indicates existence of that word, otherwise ‘0’. Then, we calculate the cosine similarity between the feature vector of the new review with the feature vector of each variety. We classify the wine to the variety whose feature vector has the highest cosine similarity with its review. The test accuracy of this model is 0.466013291831, which is a high improvement compared to baselines. We can optimize this model by including more frequent words for each variety. Instead of getting 10 words, 50 words will increase the test accuracy to 0.526247160764. We tried to include more words (100/200) but there is no substantial improvement on performance and stop around 0.54.

3.* Data Processing and Feature Extraction for other models

For models like regression and SVM, we use similar a feature set, bag of word of top 1000 most frequent words plus a price. Below is detailed description for how we process data and extract feature. To ensure our model is not biased, we randomly shuffled our dataset and then split them into 3 sets of equal size, the training set, the validation set and the testing set. Training set is for training a model. Validation set for tuning model parameter. Test set of measure model performance. The main data we will use for prediction task is the description of a review. We assume that the higher is the frequency of a word, the more important it is. Therefore we will use the top 1,000 most frequent words in the training review corpus. We observe from our explorative analysis that price of wines has moderate variance, so adding price to our feature is likely to add more expressive power to our model. We decided not to include points as one of our features because unlike price, it does not demonstrate a observable variance among data.

3.2 Model: Logistic Regression

We use the feature vector described in 3.* . We will use one-hot encoding technique to build our feature vector, 1 if the word occurs in a review; 0 otherwise. The logistic regression model we use is from a python library, sklearn. After tuning parameters using validation set. The accuracy of the logistic regression model on test set is 0.71521431, which is significant performance improvement compared to cosine similarity model. The strength of a logistic regression is that it is super fast to train, even when we have 20,000 training data. The weakness of logistic regression is that it does not directly optimize the number of misclassified points. This leaves room of improvement for a better linear model.

3.3 Model: Linear Support Vector Machine

For our linear SVM, we use the same feature set as we used for logistic regression. The SVM model we use is also from sklearn. After tuning parameters with the validation data, the SVM model has 0.755278876083 test accuracy, an observable improvement from logistic regression. The strength of this SVM is that it directly optimize the number of misclassified points, but it is more expensive to train.

3.4 Other Attempts and Optimization

In order to achieve a higher accuracy rate, we modified the third model, which is the linear support vector machine model.

3.4.a RBF Kernel SVM

We consider to change the linear SVM model to an rbf kernel SVM model. After we train an rbf kernel SVM (also from sklearn), we surprisingly find the accuracy drops to 0.69697989. The reason for this surprise might be that we did not find the optimal regularization factor. However, as we attempt to find a better regularization factor, our machines do not have enough computing resource (both speed and space) to finish model training.

3.4.b TF-IDF Feature Representation

We try to change the one-hot encoding representation of our feature to the tf-idf of the corresponding word, and use the same training approach. The test accuracy of using tf-idf feature representation is 0.75519475, which is only a slight improvement than using the one-hot encoding. This small improvement comes with a huge cost. Building a feature vector for one data point requires going through the entire dataset. Therefore, it takes $O(N^2)$ to build the training feature vectors, where N is the number data. Therefore, we decide not to proceed with tf-idf representation of our feature.

3.4.c Adding Price

We append the price of the wine at the end of the feature since we find price has a correlation with the variety. The resulting test accuracy is 0.75807362, which is a small but noticeable improvement. Hence, we decide to adopt price as one of our features.

3.4.d Extending Feature Set

Then, we decided to extend our feature vector. We change the number of most frequent word from 1,000 to 2,000 and 4,000 and apply the same model. The corresponding results are: 0.76757849 and 0.77328492, respectively. We did try some higher numbers like 6,000, but it appeared that our machines again do not have enough computing resource (both speed and space) to finish the model training.
At last, we finalize to a model using a linear SVM with 4000 more frequent words plus price as our feature vector. This model has the high enough performance and low enough cost to train among all models we have trained, with a test accuracy of 0.77328492.

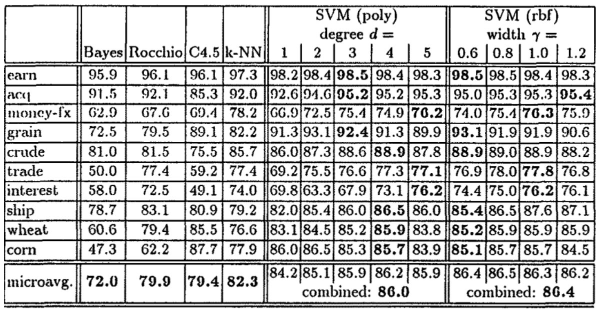
4.Relevant Literature

We obtain our dataset from Kaggle. Although we included price into our feature set, the main predictive task we are doing is still text categorization. The most commonly used dataset for text categorization is the Reuter's collection and Ohsumed collection. Reuter's collection is compiled by the Carnegie Group, Inc. (CGI) and used to evaluate one of their text categorization systems. It contains about 10,000 training data and about 3,000 testing data. Ohsumed collection is a subset of a medical database. It contains overall about 20,000 data.

Recent approach to text categorization can broken into two parts: Feature Selection Method and Model Selection. T. Joachims proves that SVM is well suited for text categorization. Y. Yang and J. Petersen compare and contrast different feature selection methods.

4.1 Support Vector Machine

In T. Joachims's experiment, he uses the Reuter's collection and Ohsumed' collection. He also uses all distinct words as his feature vector. He then compares performance of SVM (poly and rbf kernels) with that of Bayes, k-Nearest Neighbors, Rocchio and Decision Tree (Tree). Results are show below



					SVM (poly) degree $d =$					SVM (rbf) width $\gamma =$			
	Bayes	Rocchio	C4.5	k-NN	1	2	3	4	5	0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

Figure 4.1: Joachims' experiment results

From above statistics, he concludes that SVM is well suited for text categorization task.

4.2 Feature Selection Methods

The most commonly used document representation method is "bag-of-words". Because there are often tens of thousands of distinct words in a considerably large document dataset, we need some methods to measure the importance of each word, and eliminate words of low importance. We assume that the higher is the frequency of a word, the more important it is. Therefore, after taking out stop words, we select the top k most frequent words in the entire review corpus to represent our documents. However, as mentioned in Y. Ying and J. Peterson's paper, the two of state-of-art methods to measure word importance are Document Frequency(DF) and Information Gain (IG). The following section will elaborate in details about these methods and the benefit and drawback of each.

4.2.a Document Frequency (DF)

Document Frequency of a word is the number of documents in which the word appears. We then can measure the DF for each word and throw away words with a frequency lower than some predetermined threshold.

The benefit of DF is the easiness of computation. To compute the DF of a word, all we need to do to is to go through the training dataset and count the number of documents where the word appears. Therefore complexity of computing DF of a word is linear with respect the number of documents. The drawback of adopting DF is that we are dropping the low-DF words from our feature vector. This contrasts the widely received belief that low-DF words are actually more important words for category prediction. This concern should be address later with the experiement results of Y. Yang and J. Petersen

4.2.b Information Gain (IG)

Information Gain of a word measures how much information we gain with the word.

The benefit of IG is that this is a more accurate measure for the importance of a word, with respect to information theory. The drawback of using Information Gain is that it takes more resource to compute. The computation complexity of IG of a word is O(NV), where N is number of document and V is the vocabulary size.

4.2.c Correlation between DF and IG

In Y. Yang and J. Petersen experiment, they use Reuter's and Ohsumed's collection. They use kNN and Linear-Least-Squares-Fit model. Then they compare the performance of each feature selection method as they aggressively eliminate unimportant words. In general, the more unimportant words we eliminate, the better is the model performance. They discover that DF and IG are both effective feature selection method, contrast to the common doubt on the DF. In fact, they discover a strong correlation between DF and IG, shown in below image

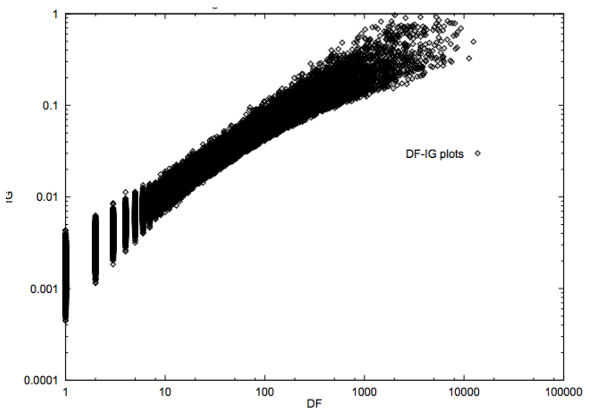


Figure 4.2: Strong Correlation between DF and IG

Therefore, despite the common belief that low-DF score words are actually relevant words for prediction, dropping words with low-DF score is an effective and efficient method for text feature selection.

5.0 Conclusion

Our final model is a support vector machine that utilizes 10 binary classifiers (one for each variety) with 4000 common words plus price as feature. Parameters of our model measure the strength of the words for predicting a category. We observe that the word with the highest weight for each category is the category name. Therefore, we conclude that the name of the category itself is most useful for separating data. This fact is verified by our baseline 2 model, where we only use the name of the category as our predicting resource and we obtain a significantly better result than only predicting the highest probability category.

The table shows that SVMs model performs moderately better than logistic regression and significantly better than cosine similarity model. And as the size of feature vector increases (more common words), the accuracy will also increase. Besides text features, 'price' does help a little bit to predict the wine variety. In both logistic regression and SVMs, expanding feature vector with more common words improves the accuracy more than just include 'price' field. We can conclude that adding more relevant words adds more predictive power to our model than price. Also, if we change the feature to tf-idf feature, the we only obtain a small improvement of test accuracy but the training time significantly longer. Therefore, we stick to the original feature.

Cosine similarity is the most intuitive model by using text features, because it predicts a document to be the most similar category. The reason that it does not

Model Used	Accuracy	Improvement over last model	Improvement over the baseline
(baseline 1)	18.7%	-	-
(baseline 2)	33.3%	⬆ 16.4%	⬆ 16.4%
Cosine similarity (50 words)	52.6%	⬆ 19.3%	⬆ 33.9%
Logistic Regression (1000 common words + price)	71.5%	⬆ 18.9%	⬆ 52.8%
Logistic Regression (1000 common words)	71.8%	⬆ 0.3%	⬆ 53.1%
SVMs (1000 common words)	75.5%	⬆ 3.7%	⬆ 56.8%
SVMS (1000 common words + price)	75.8%	⬆ 0.3%	⬆ 57.1%
RBF Kernel SVMS (1000 common words + price)	69.6%	⬇ 6.2%	⬆ 51.1%
SVMs (4000 common words)	77.0%	⬆ 1.2%	⬆ 58.3%
**SVMS (4000 common words + price)	77.3%	⬆ 0.3%	⬆ 58.6%

Figure5.1: Comparative statistics of all models we tried

perform as well as other models is that there are duplicate words in its specific feature vector. For example, “wine” appears as one of the most frequent words in four varieties ‘Pinot Noir’ ‘Red Blend’ ‘Bordeaux-style Red Blend’ and ‘Merlot’, which weakens our model because we want to build unique feature vector for each variety.

Logistic Regression is the fastest-to-train model, but it does not perform as well as SVM because Logistic Regression does not directly optimize the misclassified points.

We are surprised that the rbf-kernel SVM actually did not perform as well as the linear SVM. The reason might be that we do not have a set of optimal parameters for rbf-kernel SVM because we do not have enough computing resource to tune the model parameters.

In the end, Linear SVM with 4000 most common words plus price as feature vector is the best model we can find for this text categorization task.

References

[1] Wine Reviews Dataset on Kaggle

https://www.kaggle.com/zynicide/wine-reviews

[2] Yiming Yang, Jan O. Pedersen, *A Comparative Study on Feature Selection in Text Categorization*

http://www.surdeanu.info/mihai/teaching/ista555-spring15/readings/yang97comparative.pdf

[3] T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features.*

https://link.springer.com/content/pdf/10.1007%2FBFb0026683.pdf