BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 01/06 pp1–12
Volume 3, Number 1, March 2020
DOI: 10.26599/BDMA.2019.9020014

Classification on Grade, Price, and Region with Multi-Label and Multi-Target Methods in Wineinformatics

James Palmer, Victor S. Sheng, Travis Atkison, and Bernard Chen*

Abstract: Classifying wine according to their grade, price, and region of origin is a multi-label and multi-target problem in wineinformatics. Using wine reviews as the attributes, we compare several different multi-label/multi-target methods to the single-label method where each label is treated independently. We explore both single-label and multi-label approaches for a two-class problem for each of the labels and we explore both single-label and multi-target approaches for a four-class problem on two of the three labels, with the third label remaining a two-class problem. In terms of per-label accuracy, the single-label method has the best performance, although some multi-label methods approach the performance of single-label. However, multi-label/multi-target metrics approaches do exceed the performance of the single-label method.

Key words: classification; informatics; machine learning; multi-label; multi-target; support vector machines; wine; wineinformatics

1 Introduction

Wine has several interesting characteristics which humans enjoy: its aroma, color, and flavor being among them. Wine informatics is the study where characteristics are used to make inferences about the wine, such as its origin, its price, and its quality. Traditionally, chemical analysis would be used to represent the wine features[1–5]. Unfortunately, chemical analysis is disconnected between human perception and the features. Humans cannot intuitively appreciate a wine based on knowledge of a chemical structure. Instead, humans intuitively understand what they perceive through sensation. This can be demonstrated by a comparison of the chemical description of a wine to a qualitative description of a wine based on its review, shown in Fig. 1. As the reader can see, the chemical and fermentation details, such as the time of fermentation, the pH of the wine, and its time of harvest and bottling, do not capture the experience one has when drinking the wine. Although a review is qualitative information, there are known experts in the field of wine who make wine reviews. Many works have used wine reviews to analyze wines, and there has also been efforts to allow non-experts

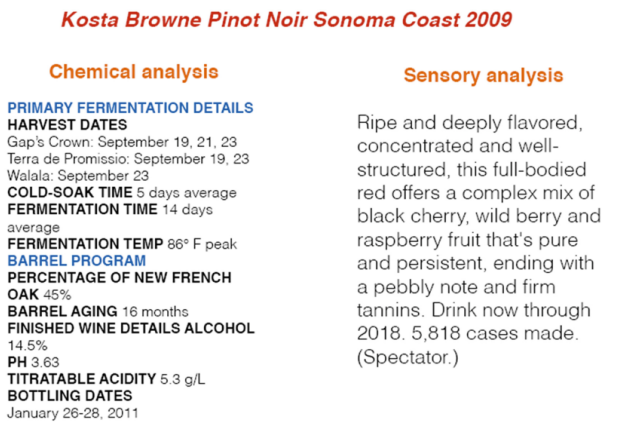


Fig. 1 An example of a wine's chemical and sensory properties. The review is passed through the computational wine wheel when constructing the key phrases[6].

• James Palmer, Victor S. Sheng, and Bernard Chen are with the Department of Computer Science, University of Central Arkansas, Conway, AR 72034, USA. E-mail: jpalmer5@cub.uca.edu; ssheng@uca.edu; bchen@uca.edu.
• Travis Atkison is with the Department of Computer Science, University of Alabama, Tuscaloosa, AL 35487, USA. E-mail: atkison@cs.ua.edu.
* To whom correspondence should be addressed.
Manuscript received: 2019-07-02; accepted: 2019-09-05

to contribute to building a qualitative description of wine[7–10]. Wine reviews may be generated by experts according to different policies. *Wine Spectator* is a popular source for wine reviews, generating reviews for about 15 000 reviews each year, and their experts follow a specific tasting guide[11]. In order to avoid bias, all of their tastings are blind tastings. Additionally, reviewers tend to focus on specific types of wines, so that they may become better experts on them. Some tastings may be made by multiple reviewers to ensure consistency and accuracy.

Based on these reviews, a representation of a wine can be constructed. In order to construct a representation of wines from these reviews, a systematic process needs to be in place. We chose to use the computational wine wheel 2.0 for this process[6, 12]. It uses reviews from *Wine Spectator* to extract important key words, such as "blackberry". Unimportant words, such as articles, are discarded. A total of 985 categories of normalized wine attributes are found in this wine wheel[6].

Using these key words extracted from the reviews that describe the wine in human language, we wish to predict the wines' price, quality, and region of origin. These are three entirely separate response variables. Normally, a prediction or classification problem focuses on determining a single variable. That variable may be a binary variable, or it may take on multiple values. When the possible values are numeric, the type of problem is regression, and when the possible values are categorical, the type of problem is classification. Additionally, if there are more than two possible values that a nominal variable may take on, the problem is a multi-class problem. When there are multiple response variables, however, the type of problem is different entirely. While the problem for each variable on its own may still be considered regression, binary classification, or multi-class classification, the whole problem with multiple response variables is known as multi-target prediction[13]. Two special cases of this problem occur when all of the response variables are categorical: multi-label and multi-target. If all of the categorical variables are binary, the problem is known as multi-label classification, otherwise it may be known as multi-class, multi-label classification, or multi-target prediction[13].

The focus of this paper is to use wine reviews to perform multi-label and multi-target classification on grade, price, and region of origin. Classification of grade based on reviews has been done before[7]. However, prediction of price and region has not been done before to the best of our knowledge. It is an interesting research avenue to consider that it may be possible to distinguish where a wine came from simply based on the wine tasting experience. It is not obvious that region can be predicted from such information. Using Fig. 1 as an example, the flavors of black cherry, wild berry, and raspberry fruit could be used by wineries all across the world. Similarly, predicting the price from this information is also interesting, as these fruits could be found in both expensive wine and cheap, poorly made wine.

2 Data

The source of the wine reviews used for this work is the website *Wine Spectator*. Wine reviews of at least good quality were pulled from the website and preprocesssed. Data with missing or indecipherable information was simply omitted. 105 085 wines and their reviews remained, ranging over a period from 2006 to 2015.

The quality of the wine is judged based on *Wine Spectator*'s 100 point scale. Quality ranging from 80–84 is good, while 85–89 is very good, 90–94 is outstanding, and 95–100 is classic[14]. Out of the entire dataset, the computational wine wheel identified 799 important key phrases (some key words were actually two or three words, such as "black cherry"). Each phrase is considered an attribute. For each phrase, an instance is assigned either a "1" if that phrase appears in its review or a "0" otherwise.

This work aims to use these attributes that describe the wine to predict the region of origin, grade, and price. To make the task both easier and more successful, some transformations to these variables were made. First, consider the region of origin. There are two widely recognized styles of producing wine: an old world style and a new world style[15]. Historically, these styles tended to reflect where a wine was made as well as the techniques used to make it. Old world wines come primarily from Europe and Israel. New world wines come from anywhere else. Sometimes North Africa and parts of Asia may also be included in this group, but sometimes they are not. For the purpose of this work, these regions are considered new world. Today, this trend largely still holds true, although there are some exceptions. Old world techniques may be applied in new world regions, and vice versa. For simplicity,

this work assumes that the historical trend is valid. We then used two classes for region of origin: If a wine originates in the old world, a value of "1" is assigned, otherwise a "0" is assigned.

We also changed grade and price from regression problems to classification problems. Specifically, we used the aforementioned grade ranges to categorize the wines into categories "good", "very good", "outstanding", and "classic", respectively. This classification scheme seems like a natural choice since *Wine Spectator* already uses it. For the price label, all prices were normalized to U.S. dollars per 750 mL, which is a common volume for a bottle of wine. Then, the quartiles were found and the wines were categorized into these quartiles: <\$18, \$18–\$29, \$29–\$50, and >\$50. Quartiles were chosen to balance the wines evenly between price groups. The final dimension of the dataset is 105 085 instances, 799 binary attributes, and 3 labels (or 105 085 rows and 802 columns), where two of the three labels are multi-class labels. The boxplots of the distribution of grades and price are shown in Fig. 2. While the grades are fairly evenly distributed, there are many price outliers. As Table 1 shows, there is also a positive correlation between the grade classes and the price of the bottle of wine.

In addition to the multi-class problem, we also use

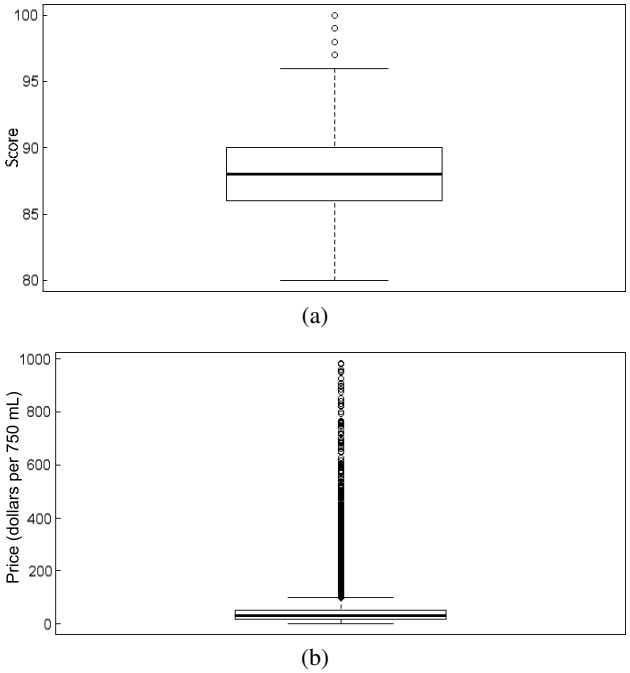


Fig. 2 (a) The boxplot of the grades. Most of the wines are in the "very good" range. (b) The boxplot of the prices. Most wines cost less than \$50 per 750 mL, but there are many expensive outliers.

Table 1 Percentage of wines which are relatively cheap or expensive for each grade category. There is a moderate positive correlation between grade and price. Price is per 750 mL bottle.

Grade category	Percent under \$50 (%)	Percent over \$200 (%)
Good	95.89	0.02
Very good	88.09	0.14
Outstanding	54.53	2.82
Classic	15.51	23.39

a modified dataset containing the same number of instances, binary attributes, and labels, but with two classes each for grade and price. These two classes are formed using the lower two and upper two classes. For grade, the "good" and "very good" classes are merged into one class, and "outstanding" and "classic" are merged into the other class. These classes may also be known as "90–" and "90+", respectively. For price, the lower two quartiles are merged into one class and the upper two quartiles are merged into the other class. These classes may also be known as "\$29–" and "\$29+", respectively. The breakdown of these classes for each dataset is shown in Fig. 3. This will allow us to compare results when there is and is not a multi-class problem. We will call this dataset the "two-class" dataset for simplicity, while the dataset with four classes for grade and price will be called the "four-class" dataset, even though there are still only two classes for region.

3 Multi-Label Approaches

First, we classified each of these dependent variables separately. This will serve as a baseline point of comparison. Next, we consider all of the labels together,

Category	Grade	Price
1	$\leqslant 84$	$\leqslant$ \$18
2	85–89	\$18–\$29
3	90–94	\$29–\$50
4	95–100	>\$50

(a) Four-class dataset

Category	Grade	Price
1	< 90	$\leqslant$ \$29
2	$\geqslant 90$	$>$ \$29

(b) Two-class dataset

Fig. 3 (a) Response variables and their class categories for the four-class dataset. Note that region remains two classes. (b) Response variables and their class categories for the two-class dataset.

thus creating a multi-label problem.

There are two general approaches to solve the multi-label problem: problem transformation and algorithm adaptation. Problem transformation applies transformations to the dataset so that the multi-label problem becomes one or more single-label problems[16]. Algorithm adaptation is where the algorithm itself is able to handle the multi-label problem directly. It turns out that many, though not all, of the algorithm adaptation methods actually use problem transformation within them[17]. Therefore, most of the discussion on multi-label methods will revolve around how the data is transformed.

3.1 Naive approaches and binary relevance

There are several problem transformation methods. The simplest and least useful one is to remove all instances where there are multiple labels, creating a single-label problem with much information loss. Because all wines have multiple labels, this method cannot be used for this problem. The second simplest method is to treat each label independently, which we did as a baseline. If the data is split up correctly, this method can also be called Binary Relevance (BR). This method is effectively the traditional classification problem, although it applied multiple times to cover all of the labels.

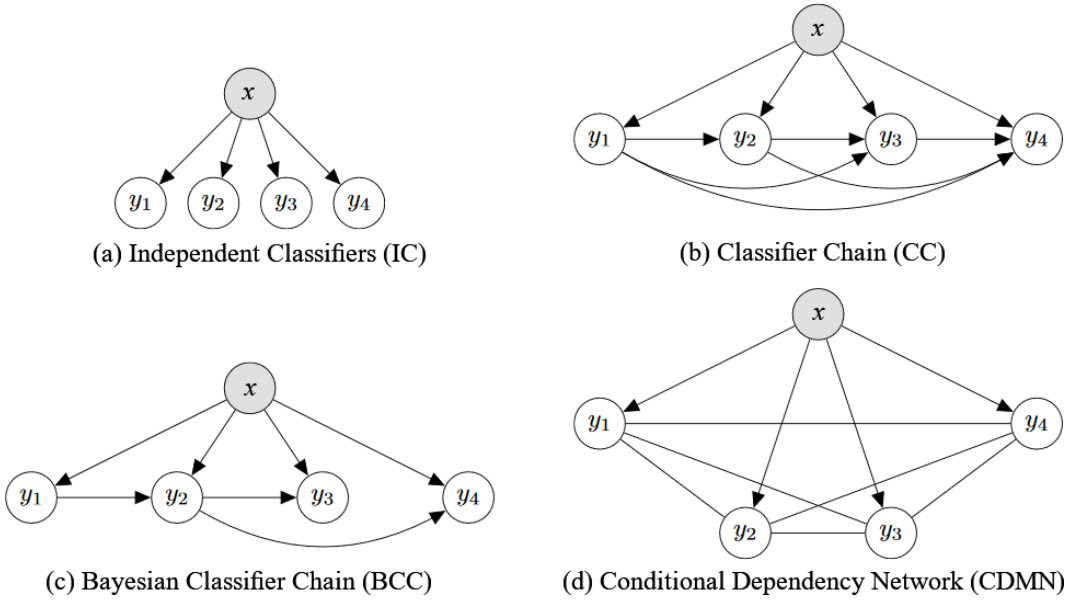
In the binary relevance method, the original dataset is split up into several datasets[17]. Each dataset has all of the instances from the original dataset and their attributes. What makes these datasets different from each other is that each dataset is going to correspond to a label. Thus, in addition to the columns needed to represent the instance attributes, there will be one column to hold the information about that particular label. For this case, that means there are 800 columns in each of these split datasets: 799 for the wine review attributes, and 1 for the label. The value in the label column will be a "1" or a "0", where a "1" means that instance has that label, and a "0" means that instance does not have that label. One dataset each will correspond to the region, price, and grade label. If we consider region, for example, this dataset has the 799 attribute column plus a column for the "old world" label. Every wine that is "old world" has a "1" in this column, and every wine that is not has a "0" in this column. Because a wine can either be "old world" or "new world", there is no need to have a separate dataset for the new world wines. However, there are four classes each for price and grade. Rather than using binary relevance, a generalized approach must be used, called class relevance. Rather than training a binary classifier on each label, a multi-class classifier is trained on the multi-class labels. Because the binary/class relevance methods essentially transform the data into several single-label problems, the performance between the baseline method and binary relevance should be comparable.

3.2 Label powerset

The second problem transformation method is the Label Powerset (LP) method[16]. In this method, a new label is created. The value of this label is one from a range, where the range is based on the number of combinations of labels. An old world wine of outstanding quality that costs $51 would be given a class value that corresponds to "old world and outstanding and price quartile four", where a new world wine of outstanding quality that costs $12 would be given a class value that corresponds to "new world and outstanding and price quartile one". The total number of classes is equal to the number of classes from each dependent variable, multiplied together. In the two-class problem for grade and price, each label is two classes, so there would be a total of eight classes. Having four classes each for grade and price makes the problem more difficult, with 32 possible combinations. A multi-class classifier is then trained on this new label.

3.3 Classifier chains

The third problem transformation method, Classifier Chains (CC), attempts to preserve dependencies between the dependent variables while still using binary relevance[18]. In the first stage of a classifier chain, a regular classifier is built for one dependent variable, like in a normal single-label problem. This classifier will make predictions for what the dependent variable should be for any given instance. In the second stage of the classifier chain, the first stage predictions are used as an additional input to predict the second dependent variable[18]. Thus, the second stage classifier uses both the attributes and the first dependent variable to make predictions about the second dependent variable. The third stage classifier would use the attributes, the first dependent variable, and the second dependent variable to make predictions about the third dependent variable, and so on. A visual of the classifier chain compared to single-label and other chain-like multi-label methods are shown in Fig. 4.



(a) Independent Classifiers (IC)

(b) Classifier Chain (CC)

(c) Bayesian Classifier Chain (BCC)

(d) Conditional Dependency Network (CDMN)

Fig. 4 A graphical comparison of different multi-label methods[20]. Here, x are the attributes, and each label is y_n.

How this operates depends on if one is training or testing. In both training and testing, the first stage of the classifier chain trains as you would expect for a normal single-label problem. In training but not in testing, the second stage classifier uses the known, correct values of the first label. In testing, the second stage classifier takes the testing output from the first stage classifier to fill in the values of the first label. Thus, while in training it would appear not to matter what order the classifier chain is in, in testing it becomes obvious that the order does matter. While there are ensemble methods to determine the correct order, the dataset is too large for this to be computationally feasible for the amount of resources we have to work on this problem. Thus, we will simply choose a descending order of prediction accuracy for the classifier chain. That is, the label we can independently predict most accurately in a single-label setup will be in stage 1, the second most accurate in stage 2, and so forth. To compare the performance of this order, we will also choose one other order at random.

3.4 Other chain-like methods

Bayesian Classifier Chains (BCCs) are another classifier chain method[19]. The basic difference between the BCCs and CCs is that BCCs are not fully parameterized while CCs are fully parameterized[20]. That is, BCCs do not necessarily use all the labels in the previous stages to classify the next staged label. The sparsity in the connections between labels leads to a faster classifier, and potentially a more accurate one, as labels which do not have dependencies between them need not be linked[20].

Another approach uses the Classifier Trellis (CT)[20]. A CT forms a directed graph between the labels in a trellis structure, where no directed loops can exist[20]. The structure is defined and fixed at the beginning of the problem, so that the computational complexity involved with finding the right structure in CCs does not occur with CTs. In the trellis, shown in Fig. 5, label dependence goes from the top left to the bottom right. Since this structure is fixed, one only needs to decide which labels to put where in the trellis. This is accomplished using a label-frequency based pairwise heuristic[20]. A random label is initially placed in the upper left corner of the trellis[20]. After that, the heuristic is used to place the remaining labels. Classification then takes place as it does with classifier chains, except that the dependency structure is based on the trellis and not a chain. An example of a classifier trellis on nine labels is shown in Fig. 5. The classifier trellis is designed to be computationally efficient even when handling large numbers of labels. Because it is optimized for performance at a large scale, we may not see performance increases on our dataset, which has only three labels.

The last approach we used is Conditional Dependency Networks (CDNs)[21]. A CDN is made by forming an undirected, fully connected graph between the labels, shown in Fig. 4. Unlike classifier chains,

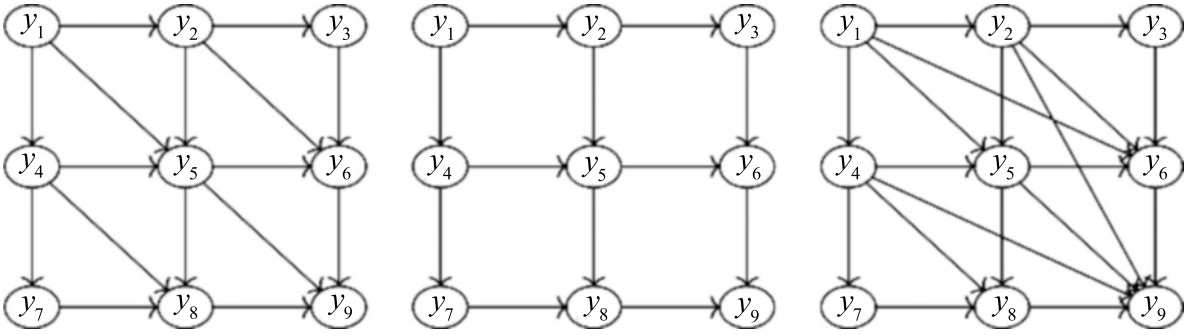


Fig. 5 Three possible trellis structures for nine labels[20].

CDNs do not try to approximate the dependence of one label on another by using several probabilistically correct classifiers chained together[20]. Instead, CDNs encode the dependencies between the different labels. This is done by training as many binary classifiers as there are labels, where each classifier defines a conditional probability distribution on one label given all the other labels and the input attributes[21]. Thus, there is a conditional probability distribution built for each label[21].

4 Evaluation Metrics

Unlike single-label data, multi-label data may have different degrees of how many labels are contained within them[17]. The two metrics for determining this are label cardinality and label density. The first metric is the average number of labels in the instances. The label density is the label cardinality divided by the total number of labels. Datasets with the same cardinality but different density may behave differently. For example, a dataset with higher density suggests that there are more instances assigned to more items than a dataset with lower density; therefore, it may be harder to predict every single label that an instance has. It could also mean that there is less variance within the data, which could in turn lead to easier predictions. Regardless, not all multi-label datasets are created equal, and care should be taken when comparing the performance of classifiers accordingly.

Since every instance has a constant number of labels, there is no need to worry about predicting the correct number of labels for an instance. Instead, predicting the right labels for that instance is the only thing this problem is concerned with.

With an understanding of how data may have different degrees of how multi-label they are, it makes sense to now consider the model evaluation metrics. The first model evaluation metric we use is Hamming loss, which is the analog to error in a single-label problem[16]. It is defined as

$$HammingLoss = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \Delta Z_i|}{|L|} \tag{1}$$

where D is the multi-label dataset, Y_i is the true label set of the i-th observation in D, Z_i is the predicted label set of the i-th observation in D, and L is the set of possible labels[17]. For each instance, Hamming loss measures the symmetric difference between the predicted label set and the true label set, normalized based on the number of labels. The lower this difference, the better the performance[16].

Another loss function is zero-one loss, which is a measure based on exact matches between the true and predicted label sets. For each observation, this measure compares both the predicted label set and the true label set. If it is not an exact match, the loss for that observation is one. If it is an exact match, the loss is zero. Specifically, zero-one loss is defined as

$$ZeroOneLoss = \frac{1}{|D|}\sum_{i=1}^{|D|}\begin{cases}1, & Y_i = Z_i;\\ 0, & Y_i \neq Z_i\end{cases} \tag{2}$$

Zero-one loss is clearly a stricter metric than Hamming loss. It is suitable for when a model models dependence between labels, where Hamming loss is not directly improved by a model which incorporates label dependence[22].

Accuracy can also measure model performance. There are two versions of accuracy: overall accuracy and per-label accuracy. Overall accuracy, also known as Jaccard index, is defined as

$$Accuracy = \frac{1}{|D|}\sum_{i=1}^{|D|}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{3}$$

The Jaccard index calculates the average of the number of times that the true and predicted labels for

an instance intersect divided by the number of true and predicted labels for that instance. In other words, for each observation, this measures how many labels are common between what is predicted and what is true, then divides that by how many different truths and predictions there are. Overall accuracy may also be given a forgiveness factor by raising this ratio to some exponent other than 1. This allows for small errors to not be penalized as much and may be useful if there are a large number of labels. It should be noted that this measure of accuracy is not directly related to loss, unlike in a single-label problem where accuracy and error are directly related.

Precision and recall can also be used to evaluate a multi-label problem. Precision measures, on average, how much is in common instances between the predicted set and true label set divided by the size of the predicted set, where recall measures, on average, how much is in common between the predicted set and true label set divided by the size of the true label set. The numerator in both of these metrics, like accuracy, counts the number of correctly classified labels for an instance[17]. In precision, the denominator counts the number of predicted labels. Thus, by dividing the number of correctly classified labels by the number of predicted labels, precision measures how many of the predicted labels were relevant or useful. In recall, the denominator counts the number of true labels for an instance. By dividing the number of correctly classified labels by the number of true labels, recall measures how many of the correct labels the classifier actually selected. The F_1 score combines both precision and recall into a single metric by taking the harmonic mean of both, shown by this formula:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4}$$

Other metrics, such as coverage and ranking loss are useful where labels are ranked or the classifier outputs confidence for a specific label[16]. This problem has no ranked labels, and there is no consideration of confidence, so these metrics are not used here.

5 Implementation and Results

5.1 Implementation

We used Support Vector Machines (SVMs) for the classification algorithm. SVMs work by constructing a hyperplane that separates classes with some margin[23]. The data points, which are used to represent this hyperplane, and therefore form the model itself, are the support vectors. Historically, SVMs have shown good results in other domains such as credit rating, drug/non-drug classification, groundwater level prediction, and image classification[24–27]. Additionally, SVMs have been used in wineinformatics to classify red wines based on their physiochemical properties[1]. It is for these reasons that we chose to use the support vector machine for this research.

There are several SVM implementations available. One popular implementation is LIBSVM, which is written in C++ and Java, and was developed by Chang and Lin[28]. LIBSVM has interfaces in over 30 programming languages and tools, making it easy to access almost any programming environment[28]. LIBSVM is an excellent choice for both its maturity and wide portability. For the single label approach, we used the LIBSVM implementation from within WEKA (the Waikato Environment for Knowledge Analysis)[29]. For the multi-label approach, we used LIBSVM from within MEKA (a Multi-Label Extension for WEKA)[30]. In all cases, C-classification was used, the cost of the constraint violation parameter was set to one, and five fold cross validations were employed.

The performance of the classifier also depends on the kernel chosen to use for the SVM. We chose to use the linear and Radial Basis Function (RBF) kernels as they provided the best results. Using the polynomial kernels with degrees 2 and 3 (quadratic and cubic kernels) performed worse, so we will not report them in the results. A comparison between the kernels for one multi-label approach is shown in Fig. 6a, while the same comparison for the single-label approach is shown in Fig. 6b.

The multi-label evaluation metrics were provided by MEKA. However, MEKA did not provide Jaccard index or F_1 score for the multi-target problem. Therefore, these metrics are omitted in our results for these experiments. Additionally, WEKA could not provide the multi-label evaluation metrics because it is inherently single-label.

A final limitation to note is that MEKA does not have multi-target extensions for every multi-label algorithm. For this reason, some algorithms cannot be applied to the multi-class multi-label problem without significant performance penalty. Label powerset should not need a multi-target extension, as the additional class values would only result in an increased number of classes that label powerset can form. However, neither the classifier

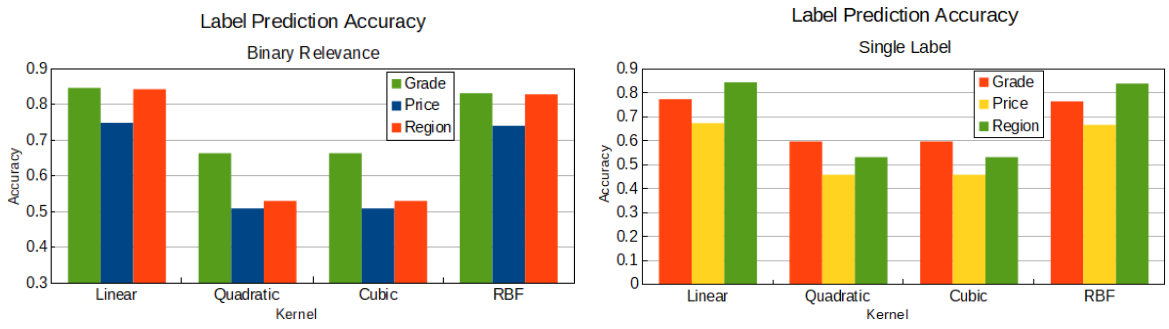


Fig. 6 (a) The multi-label accuracies for binary relevance using the two-class dataset. The quadratic and cubic kernels perform poorly. (b) The results for single-label accuracy using the two-class dataset.

trellis nor the conditional dependency network has multi-target extensions, either. The results for these algorithms will still be reported, with the caveat that any deficiencies in the accuracy may be caused by the implementation and not the method.

Several of the multi-label and multi-target implementations have parameters. Except where noted here, we used the defaults (or when only one value was appropriate for this dataset, that value was used instead). For classifier chains, the random seed was used to control the chain order. For conditional dependency networks, we reduced the number of iterations from 1000 to 100 to reduce the runtime by several days.

5.2 Two-class results

5.2.1 Single-label results

We use the single-label approach as a baseline. The results are good: Using the linear kernel, accuracies of 84.4%, 85.9%, and 74.9% for region, grade, and price, respectively were achieved. Consider that if random chance were to be used, we would expect per-label accuracies of 50% each. The RBF kernel does not perform as well with an accuracy of 82.9%, 84.9%, and 74.1% for region, grade, and price, respectively.

5.2.2 Multi-label results

In all of the multi-label methods, the linear kernel performed the best, with the RBF kernel closely following. Only considering per-label accuracies, there were only 5 cases where the multi-label approach would tie or beat the single-label accuracy. These cases are the binary relevance linear kernel on the price label, the binary relevance RBF kernel on the price and region labels, the Bayesian classifier chain linear kernel on the price label, and the Bayesian classifier chain RBF kernel on the price and region labels. While more details are provided in the following sections, the reasons for this performance depend on the method. Binary relevance, for example, is expected to perform on par with the single-label method because it is essentially a single-label method. Both label powerset and Bayesian classifier chains allow for information to be combined from the other labels, thus providing more information to the model and allowing for superior results. This is particularly useful since grade and price are moderately correlated with each other. The results are shown in Table 2.

Binary relevance. Because of the way that binary relevance does the problem transformation, we expect the results to be similar to the single-label results. Indeed, the two-class results remain competitive for two of the three labels, only suffering 1.2% degradation in accuracy for grade. This degradation is shared with all the multi-label methods except for label powerset, which performs better, and conditional dependency networks, which perform worse.

Label powerset. Of all the multi-label methods, label powerset performs the best, outperforming the single-label approach in price and region when using the linear kernel. The linear kernel label powerset performs the best in all metrics versus all other tried multi-label methods and it is the only classifier to outperform the linear single-label method in price and region. Because label powerset transforms the problem from a multi-label problem into a multi-class problem, the classifier is able to use correlating information from all three dependent variables at the same time. For example, if a particular independent variable is well correlated with grade and would predict a high quality wine, the LP approach is able to use these moderate correlations between price and grade to nudge that wine prediction towards a higher price, even if the other information about that wine is only weakly correlated to price

Table 2 Results for the two-class dataset. The best multi-label method is the linear kernel label powerset, beating all the other linear kernel methods, except for in grade versus the single-label approach.

Method	Kernel	Grade	Price	Region	Hamming loss	0/1 loss	Jaccard index	F_1
Single	Linear	0.859	0.749	0.844				
	RBF	0.849	0.741	0.829				
BR	Linear	0.847	0.749	0.843	0.187	0.451	0.676	0.787
	RBF	0.832	0.741	0.829	0.199	0.472	0.655	0.768
LP	Linear	0.849	0.753	0.851	0.182	0.423	0.691	0.794
	RBF	0.824	0.736	0.824	0.205	0.458	0.654	0.760
CC	Linear	0.847	0.732	0.844	0.193	0.452	0.665	0.779
G, P, R	RBF	0.832	0.722	0.830	0.206	0.474	0.641	0.758
CC	Linear	0.847	0.733	0.843	0.197	0.452	0.665	0.779
R, G, P	RBF	0.832	0.723	0.829	0.205	0.474	0.642	0.759
BCC	Linear	0.847	0.749	0.843	0.187	0.451	0.677	0.787
	RBF	0.832	0.742	0.829	0.199	0.471	0.656	0.769
CT	Linear	0.847	0.732	0.844	0.192	0452	0.665	0.779
	RBF	0.832	0.722	0.829	0.206	0.474	0.641	0.758
CDN	Linear	0.837	0.725	0.844	0.198	0.458	0.653	0.766
	RBF	0.812	0.710	0.830	0.216	0.486	0.624	0.737

directly.

Classifier chains, CT, and CDN. Because classifier chains depend on order, we tested two different chain orders. The first chain order is Region, Grade, Price (R, G, P). The other order we compared is Grade, Price, Region (G, P, R). Interestingly, there was not a significant decrease in accuracy between different orders. This is likely because there is a moderate relationship between grade and price. In addition, the data has a high dimension in number of attributes, so the influence from a single extra dimension added on at each stage of the chain is dwarfed by the influence of all the other dimensions. Also, it is interesting to note that the performance in price was less than binary relevance or label powerset, while the other two labels remained competitive.

The Bayesian classifier chain improved on the classifier chain was able to maintain performance in all three labels. Because of this, BCC was able to outperform the standard classifier chain in the multi-label metrics. Also, the order of the chain did not matter: The algorithm was able to identify the correct chain order regardless of our settings, and the performance was identical no matter the settings we used for chain order.

The classifier trellis has performance nearly identical to the classifier chain, even barely outperforming it in Hamming loss, while the conditional dependency network performs worst of all the tried methods. Conditional dependency networks remain competitive on the region label, but suffer in the other two labels.

5.2.3 Summary

Multi-label methods tend to be less successful than the single-label method for this problem, with the exception of the label powerset method. Region tends to be the easiest label to predict, with accuracies between 82.9% and 85.1%. Wine quality is similar to region, although a bit more challenging for multi-label to predict. Wine price is the hardest to predict, with accuracies ranging from 71% to 74.9%. Despite single-label besting most other approaches, the label powerset is able to outperform the single-label approach.

5.3 Four-class results

We now compare the results when using the dataset with two classes for region and four classes each for the grade and price labels. When multi-target extensions to the multi-label algorithms exist, the multi-target results are reported instead. Otherwise, the multi-label results are reported, often with reduced accuracy. The results are shown in Table 3.

5.3.1 Single-label results

As with before, the linear kernel performs best: 84.4% accuracy for region, 75.1% accuracy for grade, and 47.1% accuracy for price. For comparison to random chance, we would expect there to be 25% accuracy for grade and price, and 50% accuracy in region.

5.3.2 Multi-target results

In all of the multi-label methods, the linear kernel still performed the best. This is to be expected, considering

Table 3 Results for the four-class dataset. The best multi-label method is the linear Bayesian classifier chain method, beating all other methods in per-label accuracy and Hamming loss. BCC achieves fourth place (considering ties) in 0/1 loss.

Method	Kernel	Grade	Price	Region	Hamming loss	0/1 loss
Single	Linear	0.752	0.470	0.844		
	RBF	0.702	0.448	0.829		
CR	Linear	0.752	0.470	0.844	0.312	0.697
	RBF	0.702	0.449	0.829	0.340	0.737
LP	Linear	0.583	0.298	0.849	0.423	0.818
	RBF	0.544	0.247	0.821	0.463	0.864
CC	Linear	0.752	0.463	0.843	0.314	0.693
G, P, R	RBF	0.702	0.432	0.829	0.346	0.738
CC	Linear	0.752	0.464	0.844	0.314	0.692
R, G, P	RBF	0.702	0.435	0.829	0.344	0.736
BCC	Linear	0.752	0.473	0.844	0.311	0.694
	RBF	0.702	0.449	0.830	0.340	0.736
CT	Linear	0.752	0.463	0.843	0.314	0.693
	RBF	0.702	0.432	0.829	0.346	0.738
CDN	Linear	0.739	0.459	0.843	0.319	0.694
	RBF	0.680	0.420	0.830	0.357	0.744

the results for the two-class dataset are better with a linear kernel.

Class relevance. Class relevance is the multi-target extension to binary relevance. This approach performs identically in per-label accuracies when compared to the single-label approach. It has average Hamming loss, but the second worse zero-one loss.

Label powerset. Unfortunately, label powerset suffers degraded performance in grade and price, but remains competitive in region. This is due to the increased number of classes that must be formed from the greater number of label combinations found in the four-class dataset. When there are more classes, the probability of error increases. While LP performed the best with two classes, it now performs the worst with multiple classes. Even so, it has the highest performance in predicting region.

Classifier chains, CT, and CDN. Classifier chains continue to perform well when given the multi-target dataset. Like in the two-class problem, the order of the classifier chain shows some, but minor, differences, with a chain order of region, grade, and price beating the chain order of grade, price, and region in zero-one loss only by tenths of a percent.

The Bayesian classifier chain was able to outperform the regular classifier chain, like in the two-class problem. Based on the results of our research, the order of the chain set by parameters has little influence in this dataset. Although the zero-one loss is slightly higher compared to regular classifier chains, the per-label accuracy and Hamming loss are improved.

Both the classifier trellis and conditional dependency network have average results compared to the other methods, with the classifier trellis performing better between the two of them. Although not as good as the classifier chain in per-label accuracies, the classifier trellis got very close in Hamming loss and zero-one loss, especially with the linear kernel. Despite the CDN having a poorer Hamming loss, it is able to remain competitive in zero-one loss.

5.3.3 Summary

The single-label approach still remains difficult to beat. While most algorithms are able to tie its performance in grade and region, the same is not true for price. In our case, the Bayesian classifier chain is best able to scale up to the multi-target approach compared to the other methods we tried. It achieves the best Hamming loss, ties the single-label approach in grade and region, and outperforms the single-label approach in price.

6 Conclusion and Future Work

Treating each label independently is usually the most successful approach to working with this data when considering per-label accuracy. This is to be expected when each label is important on their own, and no care is given to getting multiple targets correct simultaneously. Although it is difficult for the multi-label/ multi-target approach to tie, let alone outperform, the single-label approach in per-label accuracy, the Bayesian classifier chain is able to do this for both of

our datasets. While label powerset performs the best for the two-class dataset, the BCC remains the most reliable overall. Even though treating each one of these targets individually might yield better performance for that target, considering them together can lead to a better overall result.

We have shown that it is possible to simultaneously classify where a wine originated, how good it is, and how expensive it is using keywords extracted from its review. This approach has the advantage of avoiding the expense and less intuitive interpretation that comes with chemical analysis. In the future, we intend to solve some of the limitations in this work. In particular, we will use regression instead of classification on price and grade. This will result in more granular single- and multi-target models. We hope that this and future work will be eventually used to improve the buying experience for wine consumers.

References

[1] Y. Er and A. Atasoy, The classification of white wine and red wine according to their physicochemical qualities, *Int. J. Intell. Syst. Appl. Eng.*, vol. 4, pp. 23–26, 2016.

[2] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, Modeling wine preferences by data mining from physicochemical properties, *Decis. Support Syst.*, vol. 47, no. 4, pp. 547–553, 2009.

[3] S. E. Ebeler, Linking flavor chemistry to sensory analysis of wine, in *Flavor Chemistry: Thirty Years of Progress*, R. Teranishi, E. L. Wick, and I. Hornstein, eds. Boston, MA, USA: Springer, 1999, pp. 409–421.

[4] S. Chung, T. S. Park, S. H. Park, J. Y. Kim, S. Park, D. Son, Y. M. Bae, and S. I. Cho, Colorimetric sensor array for white wine tasting, *Sensors*, vol. 15, no. 8, pp. 18197–18208, 2015.

[5] J. Fu, C. Q. Huang, J. G. Xing, and J. B. Zheng, Pattern classification using an olfactory model with PCA feature selection in electronic noses: Study and application, *Sensors*, vol. 12, no. 3, pp. 2818–2830, 2012.

[6] B. Chen, C. Rhodes, A. Yu, and V. Velchev, The computational wine wheel 2.0 and the TriMax triclustering in wineinformatics, in *Proc. 16^{th} Industrial Conf. Data Mining*, New York, NY, USA, 2016, pp. 223–238.

[7] B. Chen, V. Velchev, B. Nicholson, J. Garrison, M. Iwamura, and R. Battisto, Wineinformatics: Uncork Napa's cabernet sauvignon by association rule based classification, in *Proc.* 2015 *IEEE 14^{th} Int. Conf. on Machine Learning and Applications*, Miami, FL, USA, 2015, pp. 565–569.

[8] B. Chen, H. Le, C. Rhodes, and D. S. Che, Understanding the wine judges and evaluating the consistency through white-box classification algorithms, in *Advances in Data Mining. Applications and Theoretical Aspects*, P. Perner, ed. Springer, 2016, pp. 239–252.

[9] N. Wariishi, B. Flanagan, T. Suzuki, and S. Hirokawa, Sentiment analysis of wine aroma, in *Proc. 2015 IIAI 4^{th} Int. Congress on Advanced Applied Informatics*, Okayama, Japan, 2015, pp. 207–212.

[10] B. Flanagan, N. Wariishi, T. Suzuki, and S. Hirokawa, Predicting and visualizing wine characteristics through analysis of tasting notes from viewpoints, in *HCI International 2015–Posters' Extended Abstracts*, C. Stephanidis, ed. Springer, 2015, pp. 613–619.

[11] Wine Spectator, About our tastings, http://www.winespectator.com/display/show/id/about-our-tastings, 2018.

[12] B. Chen, C. Rhodes, A. Crawford, and L. Hambuchen, Wineinformatics: Applying data mining on wine sensory reviews processed by the computational wine wheel, in *Proc. 2014 IEEE Int. Conf. on Data Mining Workshop*, Shenzhen, China, 2014, pp. 142–149.

[13] E. Spyromitros-Xioufis, W. Groves, G. Tsoumakas, and I. Vlahavas, Multi-label classification methods for multi-target regression, arXiv preprint arXiv: 1211.6581, 2012.

[14] Wine Spectator, Wine Spectator's 100-point scale, http://www.winespectator.com / display / show / id/scoring-scale, 2018.

[15] K. Anderson, *The World's Wine Markets: Globalization at Work*. Cheltenham, England: Edward Elgar, 2004.

[16] C. A. Tawiah and V. S. Sheng, Empirical comparison of multi-label classification algorithms, in *Proc. 27^{th} AAAI Conf. on Artificial Intelligence*, Bellevue, WA, USA, 2013, pp. 2–6.

[17] G. Tsoumakas and I. Katakis, Multi-label classification: An overview, in *Database Technologies: Concepts, Methodologies, Tools, and Applications*, J. Erickson, ed. Barcelona, Spain: IGI Global, 2009, pp. 4–6, 10–12.

[18] J. Read, B. Pfahringer, G. Holmes, and E. Frank, Classifier chains for multi-label classification, *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, 2011.

[19] J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga, Bayesian chain classifiers for multidimensional classification, in *Proc. 22^{nd} Int. Joint Conf. on Artificial Intelligence*, Barcelona, Spain, 2011, pp. 2192–2197.

[20] J. Read, L. Martino, P. M. Olmos, and D. Luengo, Scalable multi-output label prediction: From classifier chains to classifier trellises, *Pattern Recognition*, vol. 48, no. 6, pp. 2096–2109, 2015.

[21] Y. H. Guo and S. C. Gu, Multi-label classification using conditional dependency networks, in *Proc. 22^{nd} Int. Joint Conf. on Artificial Intelligence*, Barcelona, Spain, 2011, pp. 1300–1305.

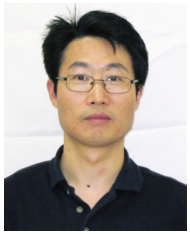
[22] J. Read, Multi-label classification, https://jmread.github.io/talks/Tutorial-MLC-Porto.pdf, 2015.

[23] D. Fradkin and I. Muchnik, Support vector machines for classification, *DIMACS Series in Discrete Mathematics and Theorectical Computer Science*, vol. 70, pp. 13–20, 2006.

[24] B. Baesens, T. van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, Benchmarking state-of-the-art classification algorithms for credit scoring, *J. Oper. Res. Soc.*, vol. 54, no. 6, pp. 627–635, 2003.

[25] E. Byvatov, U. Fechner, J. Sadowski, and G. Schneider, Comparison of support vector machine and artificial neural network systems for drug/nondrug classification, *J. Chem. Inf. Comput. Sci.*, vol. 43, no. 6, pp. 1882–1889, 2003.
[26] H. Yoon, S. C. Jun, Y. Hyun, G. O. Bae, and K. K. Lee, A comparative study of artificial neural networks and support vector machines for predicting groundwater levels in a coastal aquifer, *J. Hydrol.*, vol. 396, nos. 1&2, pp. 128–138, 2011.
[27] O. Chapelle, P. Haffner, and V. N. Vapnik, Support vector machines for histogram-based image classification, *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1055–1064, 1999.
[28] C. C. Chang and C. J. Lin, LIBSVM: A library for support vector machines, *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.
[29] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, The WEKA data mining software: An update, *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
[30] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, MEKA: A multi-label/multi-target extension to Weka, *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 667–671, 2016.

James Palmer graduated from the University of Central Arkansas with BS degree in computer science and applied mathematics in May, 2017, followed by MS degree in computer science in May, 2018. He currently works as a data scientist at Acxiom, LLC. He was the recipient of the best paper award from ICCCS'18. His interests include wine, machine learning, and big data.



Victor S. Sheng received the MSc degree from the University of New Brunswick, Fredericton, NB, Canada, and the PhD degree from the University of Western Ontario, London, ON, Canada, both in computer science, in 2003 and 2007, respectively. He is an associate professor of computer science with the University of Central Arkansas, Conway, AR, USA, and the founding director of Data Analytics Laboratory, University of Central Arkansas. After receiving the PhD degree, he was an associate research scientist and NSERC postdoctoral fellow in information systems with the Stern Business School, New York University, New York, NY, USA. His research interests include data mining, machine learning, crowdsourcing, and related applications. He has published more than 140 research papers in conferences and journals of machine learning and data mining. Most papers were published in top journals and conferences in data science, such as PAMI, TNNLS, TKDE, AAAI, KDD, IJCAI, and ACMMM. He is an SPC and PC member for many international conferences and a reviewer of more than twenty international journals. He was the recipient of the best paper award runner up from KDD'08, the best paper award from ICDM'11, the best student paper award finalist from WISE'15, and the best paper award from ICCCS'18.

Travis Atkison is an assistant professor of computer science and the director of the Digital Forensics and Control Systems Security Lab (DCSL) at the University of Alabama. He received BS degree in electrical engineering and BS degree in computer science in 1995, and MS degree in computer science in 1997, all from the University of Alabama, and received PhD degree in computer science from Mississippi State University in 2009. He has been employed with the National Security Agency, Louisiana Tech and the University of Alabama. He has authored approximately 60 peer reviewed articles in outlets such as *IEEE Transactions on Dependable and Secure Computing*, *International Journal of Critical Infrastructures*, and *IEEE Eurographics Visualization Symposium*. He has been awarded funding from multiple agencies including NSF, DOE, ALDOT, and AFOSR among others. His current research efforts focus on the topics of cyber security and transportation infrastructure. These efforts include malicious software detection, threat avoidance, digital forensics, and security in control system environments directed toward transportation. He currently holds an active CISSP (Certified Information Systems Security Professional) certification.

Bernard Chen is an associate professor of computer science at the University of Central Arkansas. He received the PhD degree from Georgia State University with bioinformatics concentration in 2008. He has authored over 70 peer reviewed journal and conference papers. His current research focuses on data science with many different application domains, including bioinformatics, wineinformatics, and coffee-informatics. He is also program committee member in many reputable international conferences, such as BIBM, ICDM, and ICMLA.