Predicting Wine Prices from Wine Labels

Jamie Ray[1,*], Svetlana Lyalina[2] and Ruby Lee[2]
[1]Department of Engineering Physics, Stanford University
[2]Department of Computer Science, Stanford University

Submitted on December 14, 2012

ABSTRACT

Motivation: The wine label is an icon that unites powerful branding and fine design under a single image. With U.S. wine consumption predicted to increase to $23.8 billion by 2014, wine labels are used not only to convey a given wine's type, producer, and origin, but also its emotional and cultural appeal. Thus, a wine label is carefully designed to convey the proper messaging of a particular bottle, whether it is for the casual consumer or the choosy collector. We have applied machine learning to the art of wine label design by learning the design features which most influence a bottle's popularity and selling price. We have collected over 30,000 wine labels and associated data on price, type, origin, and other notes from the Web. From this data, we have developed an extensive training set by extracting particular graphical features from the labels and textual features from the metadata. By implementing a support vector machine (SVM) and a Naïve Bayes classifier, we have worked to develop a preliminary model for classifying wines, based on their labels, into specific price bins. However, our results demonstrate that the features we've chosen have little power in predicting the price (bin) of a wine bottle, especially at granularities that might be useful.

1 INTRODUCTION

The wine label is an icon that unites powerful branding and fine design under a single image. Wine labels are used not only to convey a given wine's type, producer, and origin, but also its emotional and cultural appeal. Thus, a wine label is carefully designed to convey the proper messaging of a particular bottle, whether it is for the casual consumer or the choosy collector [1]. During the California wine renaissance beginning in 1966, the evolution of the wine label reflected the shift from traditional, old-world bottles to fresh, artistic wine packaging [2].

By 2014, wine consumption in the United States alone is predicted to increase to $23.8 billion [3]. Today, wine producers hire designers and consultants to craft wine labels that construct a powerful message to their target demographic, which will purchase wines within a specific price range. In order to better understand and potentially improve this design process, we have worked to develop a wine label classification model, which will predict wine prices based on both graphical features from the wine label and textual features from the wine notes, type, and origin. We hope that future wine producers will be able to use such a model to inform their design decisions.

[*]To whom correspondence should be addressed.

2 METHODS & RESULTS

Data Collection. There are a number of online wine shopping websites that have thousands of wines associated with price, origin, and producer of the bottle. These include wine.com and winechateau.com. However, the images on winechateau.com contain the entire wine bottle, while wine.com has images of the label only. Thus, we decided to scrape wine.com for all of its 30,000 wine labels and wine name, price, type, varietal, appelation, region, and additional notes. We used the wine.com API to obtain a printer-friendly version of each wine's page. We then wrote a scraper with Python and the BeautifulSoup library that locally saves the wine label and the associated data in XML format.

Feature Generation: Graphical Features. We use MATLAB's Computer Vision and Image Processing toolboxes to generate features from JPEG images of the wine labels, usually in the range of 300x300 pixels. We found that MATLAB is sufficient to compute a large number of features in reasonable time:

Image features (~.2 seconds for a 300x300 image):

- height
- width
- aspect ratio (height / width)
- background color (average over pixels in flat regions, may switch to k-means on these pixels)
- grayscale Otsu threshold (after CLAHE), and number of distinct regions generated by thresholding
- centrality of edges and corners (weighted mean with pixels towards center given higher Gaussian weights)

We also analyzed subsections of each image. These subsections are:

- red pixel values
- green
- blue
- hue
- saturation
- value
- fourier transform
- grayscale
- smaller square (quarter of full image) in center of grayscale

- edge map (Canny edge detector)
- corner strength map
- distance map (where distances are computed from nearest edge)

For each subsection, the image features are:

- mean
- variance
- quantiles ([0 .01 .05 .25 .5 .75 .95 .99 1])
- entropy
- regular Otsu threshold without CLAHE and effectiveness metric
- quantiles of gradient magnitude
- symmetry score (MSE across four axes of symmetry)
- number of blocks in quadtree decomposition of square-cropped image

The background color is computed as a mean over pixels in ‘flat’ areas of the image. We use the edge map to define flat regions as those that are greater than a certain distance away from any edges. After examination of wine labels, we find that there is often a single background color upon which text and graphics are set, suggesting that this simple method can be successful in a majority of cases. The thresholding step begins with the grayscale image, then performs an adaptive contrast-equalization step to adjust for large-scale intensity gradients across the image. This allows us to binarize the adjusted grayscale image using an intensity cutoff, determined using Otsu’s method (which minimizes the inter-class variance between the light and dark groups). We also analyze the performance of this binarization in terms of the number and size of segments it creates. Finally, we consider the possibility that image features in the center of the label may be more predictive than those towards the edges, so we calculate some features such as edge and corner strengths with greater weights for features in the center (these weights fall off as a Gaussian in x and y pixels from the center). Edge and corner values give a measure of how much structure and change is present - note that we expect many edges and corners when text is present, so there may be a correlation to the amount of text.

By subsections of each image, we mean that the original image is the RGB color representation of the wine label. This 3 color-channel image is then processed into components (each of the same size, such as 300x300 pixels) which more closely correspond to features of interest. For example, the HSV (hue, saturation, and value) color space may contain structure that isn’t easily determined from RGB analysis (like gradients in hue that cannot be recreated from processing individual color channels). We also consider features involving the frequency components of the grayscale image computed via the 2d DFT. In keeping with our hypothesis about the importance of positional information in the image (lost in calculating mean, variance, etc) we also crop the label to a small central square whose image-wide features can be compared to the original. The edge map is a binary image combining strong and weak edges found with a Canny edge detector, essentially looking for large connected, oriented gradients in the grayscale image. The corner map actually assigns values based on the gradient magnitude in different directions (strong corners should have large gradients in multiple directions). Finally, the distance map computes (Euclidean) distance from the nearest edge, and can help distinguish between images with large flat regions and those with lots of interspersed structure.

For each of these processed images, we compute a standard set of image statistics, which consider them as either 1d or 2d signals. We use mean, variance, quantiles at various percentages, entropy (measure of texture in 1d signal, related to Shannon information), an Otsu threshold without contrast adjustment, and the quantiles of gradient values (calculated at first order as the difference between adjacent pixels). We also compute mean standard error between two halves of the image, comparing top/bottom, left/right, and the diagonal symmetry of a square-cropped version. Finally, the same cropped square is decomposed into smaller squares recursively until a threshold of uniformity is met (quadtree decomposition) – so a large number of resulting squares suggests lots of texture in the image. We note that for some of the processed images (like the Fourier transform), these statistical features may not contain useful information – for example the symmetry or quadtree decomposition in frequency space doesn't have a clear aesthetic interpretation. However, it doesn't take much extra time to calculate them, keeps the code simpler, may yield some surprisingly useful features, and they can always be easily discarded.

Feature Generation: Textual Features. After binning the wine notes in $2 increments, we calculated the tf-idf (term frequency * inverse document frequency) of all terms, considering all the notes from a single bin to be a “document.” Tf-idf is high for frequent words, but is not skewed towards common low information words (“of”, ”and”, ”the”, etc.). The top ten words for each bin were collected to make up a set of words to be used as features. This was done to shrink the number of words that would be counted as features for an algorithm combining both text and image data, as opposed to blindly including all words ever encountered. The wine type, varietal, appellation, and region will also be used in the model, but these features did not require further parsing after data collection.

Model Creation & Analysis: Graphical Features. Features were generated in several (separate) runs through the list of almost 30,000 wines, including processing of the label images (570 continuous-valued features), extraction of classifiers like region and varietal (28 binary/categorical features), and text processing on the description (1200 binary features). We first needed to combine all the features (1798 total per wine) into a final dataset in the appropriate form for learning. Certain wines didn't have images or adequate descriptions, so these were discarded from our list. In addition, several of the wines were in 375 mL (half-size) bottles, which would confound the learning process in that they have the same label and description as a regular size bottle of the same wine, but roughly half the price. Finally, after plotting the distribution of prices we realized that there might not be enough statistics for wines valued in the long upper tail. As a result, we decided to discard those above the 95th percentile, and restrict our problem to learning wine pricing in the range below $110 to preserve sufficient data for training. This left us with a learning dataset of 24579 wines with 1798 features per wine.

To make features more comparable, we first normalized each feature to have zero mean and unit variance. This allowed us to easily perform Principal Component Analysis on the data, which we hoped would be useful to reduce the dimension of the problem. Fig. 1 shows a cumulative sum of the eigenvalues from the covariance matrix, demonstrating that much of the covariance can be replicated using a small fraction of the principal components. We stored the eigenvectors and eigenvalues so that we could test learning using different combinations and numbers of principal components. The principal components also suggest that no single feature or set of few features is effective in preserving covariance, as they contain non-negligible components along several tens of features, which is easily visualized by plotting. As originally formulated, the question of 'predicting wine prices' is truly a regression problem, but to begin and determine whether a solution might be feasible, we decided to start by discretizing the prices into bins. Using the coarsest possible binning, we could label wines as either high- or low-priced, using the median price as the dividing point. Another more interesting option is to use 10 bins, which we choose based on the percentiles of prices (first bin corresponding to the lowest 10% of prices, for example). For the latest classifier, data was also scaled to be in the range [0, 1] after application of PCA. This was performed both to make values easily comparable and based on the suggestion of the LibSVM algorithm's writers. Finally, in case there was some structure of the wine id's that could invalidate the assumption that the training and testing sets were equally distributed, the order of wines was randomly scrambled before assigning individual samples to be used for training or testing.

Finally, before ultimately training the desired classifier, we tested many combinations of model parameters and input features in an attempt to understand and optimize the learning process. Models included Naive Bayes (using the binary word features only) and Support Vector Machines (on all features) with a variety of objectives, kernels, and parameters. The latter used the free LibSVM compiled C code (CITE). We tested using different numbers of principal components, with and without the [0, 1] scaling, and also tried training with the full set of unprocessed data. Although in many cases the differences in performance weren't large, we converged on an SVM with the nu objective and radial basis function (gaussian) kernel, using ~100 principal components and scaling the resultant features. A set of grid searches over the model parameters (coarse then fine) suggested that we should set gamma to ~30 and C to ~3 for optimal performance (Fig. 2). The grid search was performed with 5-fold cross validation, and the learning curves generated using a training set of specified size and a default testing set of 5% of the data, or 1229 samples (Fig. 3). Other models tested included a MATLAB implementation of an SVM, and a regression-SVM built into the libSVM software, which performed poorly, with its predictions limited to the range [18, 28].

The results of training are disappointing to describe, as the classifier's performance leaves significant room for improvement. Training with the model selected as described yields a support vector machine with an accuracy of 73.35% when binary labels are generated by binning the prices into two bins. If we train with 10 labels, the accuracy is only 13.21%. Both of these accuracies were determined using five-fold cross-validation on the full dataset, and the latter case used a set of one-vs-one SVM models to choose the label with the most votes. According to LibSVM's output the optimization converged, but included a large number of the training samples as support vectors, with a large fraction of those as bounded support vectors (alpha inequalities satisfied with equality).

Model Creation & Analysis: Textual Features. For the text descriptions, we built a Naïve Bayes classifier, using the set of top scoring tf-idf words as features. Training was done on a randomly selected 70% of the data, and then the classifier was tested on the remaining 30%. Text data did not prove to be an informative source of class information, as the accuracy of the classifier generally ranged between 0.16 (MATLAB implementation) and 0.30 (Python NLTK implementation) when trying different sized bins (started with $2 bins, later settled on 10 equally populated bins with custom bounds) and different sets of indicator words (initially best by tf-idf, then highest mutual information score words, and later tried full set of all words).

We tried to apply a more advanced clustering algorithm based on dimensionality reduction (t-SNE [4]), but this showed an equally bleak picture – an amorphous cloud of points with no visible unified clusters (Fig. 4). In a final attempt to observe some form of coarse-grain structure in the text, we calculated the mutual information of each word with a particular bin. The top 15 words by mutual information for each price segment are shown in Fig 5.

3 DISCUSSION

There are multiple reasons why the Naives Bayes classifier demonstrated low accuracy: first, each wine producer allocates a wildly different amount of effort into writing the description for their product, leaving us with some wines that had essentially only the year, type and maybe location in the text, while other wines had a full paragraph describing the flavors and smells in extensive detail. In fact, when we tried clustering the wines using k nearest neighbors based on a vector of word frequencies, the only clusters that made sense were the ones where producers put in minimal effort into the description. This was not necessarily the cheapest wines, just the ones with laconic texts. Laplace smoothing was applied in an attempt to ameliorate the problem of sparse word vectors, but had little effect on accuracy.

In the list of top 15 words by mutual information for each price segment (Fig. 2), little difference can be seen in the words deemed important, with the possible exception of the bin 19.99-21.99 which has a more diverse set of words. This seems to be a cusp in the pricing landscape, with producers trying to separate themselves from their lower priced peers. Interestingly, nothing about the most expensive price segment is particularly distinctive. In fact, it shares three words with the cheapest segment.

In accordance with the large variance in wine prices and the idea that their labels and descriptions hold significant influence over the buyer's decision, it seems reasonable that with a combination of features derived from them and a large number of wines one might train a classifier to predict those wines that are more 'successful' (and should therefore be priced higher). The reasons that our approach was only marginally successful are unclear, even after significant diagnosis. It is possible that the features used don't effectively capture the qualities that make a wine appealing. Furthermore, even if they do, the directions in feature space along which such qualities occur may exhibit less variance than other less-predictive directions, which would result in predictive power being lost during PCA. The model selected may not in fact be optimal for solving this problem, as the high accuracy on training sets seen in the learning curve may be a sign of overtraining. However, the data processing and model selection described are intended to ameliorate overtraining and further fixes aren't obvious. While sanity checks were performed at various stages in the process, it is still possible that during the process of scraping prices, images, and descriptions - or while combining features from these different sources - some of the data was corrupted or mixed incorrectly, which would randomize the correlations to some extent and confound the training. Finally, one might imagine that the original problem wasn't well-formed. There are two senses in which this may be the case: first, perhaps the label and description aren't significant 'predictors' of wine price. More optimistically, the connection between 'success' and price might be too naive. Our features were chosen to separate appealing wines from less-appealing (and therefore less-successful) bottles, but it doesn't logically follow that the most aesthetically pleasing label and enticing description will be found on the most expensive wine. Rather, a better metric than price may have been the volume sold or the profit (total or per bottle). These considerations could all be useful in refining the learning process. Nevertheless, this work has demonstrated that the features we've chosen have little power in predicting the price (bin) of a wine bottle, especially at granularities that might be useful.

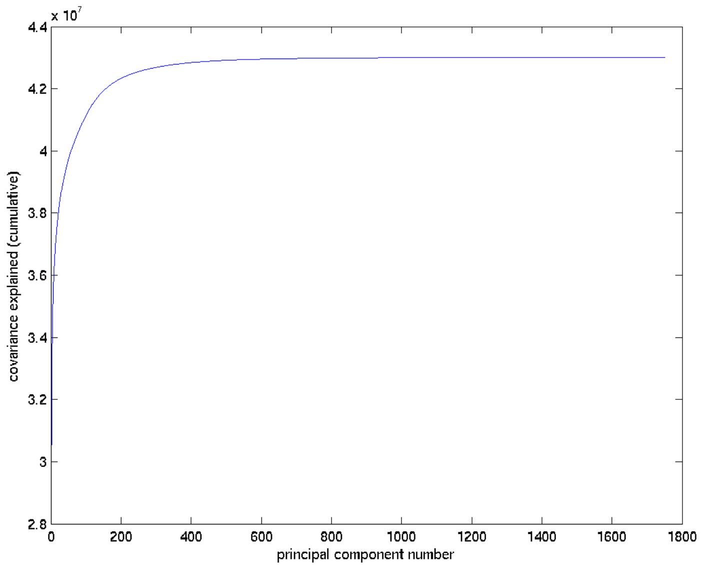
ACKNOWLEDGEMENTS

The wine label data was obtained from wine.com. Additionally, we are very grateful to the CS 229 course staff, especially Professor Andrew Ng, for their inspiration and guidance.

REFERENCES

[1] Thomas, D. A., and Pickering, D. G. (2003). The Importance of Wine Label Information. *International Journal of Wine Marketing*, *15*(2), 58–74.
[2] Caldewey, Jeffrey. Icon: The Art of the Wine Label. South San Francisco: The Wine Appreciation Guild, 2003.
[3] 2011-2012 State of the Wine Industry. Silicon Valley Bank. Web. 14 Dec 2012. <www.svb.com/2011-wine-report-pdf>.
[4] van der Maaten, L.P.J. and G.E. Hinton (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(11): 2579–2605

FIGURES



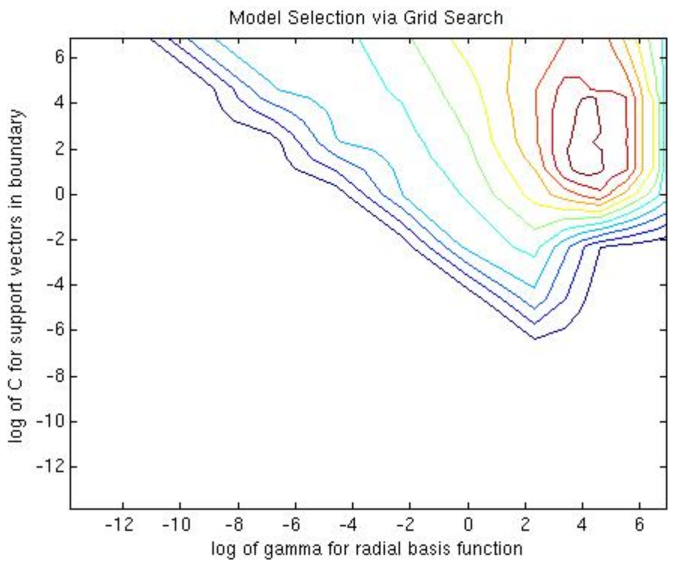
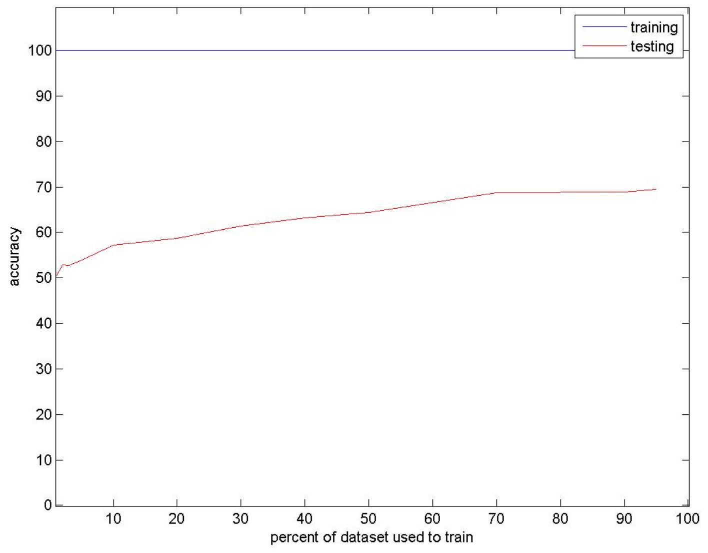


Figure 1 (left). Cumulative sum of the eigenvalues from the covariance matrix for PCA, demonstrating that much of the covariance can be replicated using a small fraction of the principal components.

Figure 2 (right). A set of grid searches over the model parameters (coarse then fine) suggested that we should set gamma to ~30 and C to ~3 for optimal SVM performance.



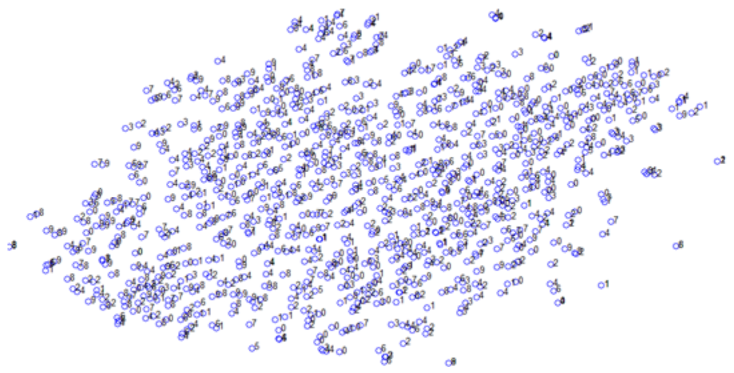


Figure 3 (left). SVM learning curves, generated using a training set of specified size and a default testing set of 5% of the data, or 1229 samples. Learning curves generated with model parameters that should be less prone to overfitting show reduction of both testing and training accuracy.

Figure 4 (right). Plot produced by t-SNE, demonstrating lack of structure in the text data. Labels are price bins as defined earlier. Randomly selected 1000 text descriptions for plotting.

0.00–10.49	10.49 – 12.99	12.99 – 14.99	14.99 – 16.99	16.99 – 19.99	19.99 – 21.99	21.99 – 27.99	27.99 – 38.99	38.99 – 54.99	54.99 – 109.99
balanced	fruit	well	great	long	powerful	grapes	aromas	full	mouth
spicy	sauvignon	fruit	balanced	fruit	meats	wines	fruit	ripe	plum
color	finish	color	fruits	fresh	soft	balanced	notes	fruits	color
well	full	grapes	valley	finish	great	sauvignon	nose	spice	finish
hints	hints	plum	mouth	well	strawberry	fruits	sauvignon	valley	grapes
full	notes	ripe	aromas	wine	integrated	hints	balanced	fruit	notes
aromas	well	fruits	notes	cherry	small	spicy	full	intense	vineyard
acidity	ripe	intense	nose	nose	silky	ripe	hints	grapes	ruby
fruit	palate	ruby	texture	tannins	lemon	flavors	acidity	finish	palate
ripe	blend	finish	chardonnay	notes	style	ruby	intense	ruby	spice
sauvignon	balanced	spice	finish	rich	estate	balance	ripe	notes	bouquet
bright	acidity	vanilla	well	flavors	cellar	full	finish	vanilla	well
sweet	color	notes	ruby	aromas	brilliant	mouth	sweet	color	texture
grapes	aromas	hints	complexity	color	grenache	great	well	well	full
blend	wine	chardonnay	bouquet	spice	aromatic	years	grapes	palate	valley

Figure 5. Top 15 words from wine description by mutual information for each price segment.