**Vintage Wine Catalogs**

**Data Extraction of Scanned Wine Catalogs from 30’s to 80’s**

Tommy Lee, Arin Sadeghi, Nir Voloshin

<https://nirvolo.wixsite.com/wine-catalog/>

<https://github.com/rin7/Wine_catalogs.git>

12 June 2018

## **Executive Summary:**

## We developed an automated method to extract key features (Appendix 5) such as the wine name, bottle and case prices from 4,111 scanned Sherry Lehmann wine catalogs from the 1930’s to 1980’s¹. By experimenting with different optical character recognition (OCR) software, we chose Tesseract (Appendix 4) to build our algorithm. Using a randomly sampled subset of 100 images, we created a training set of three catalogs to train our algorithm over time. We selected 12 catalogs as a test set which all have similar page designs because our methods are based on certain characteristics. These characteristics range from at least one price column in a page and the wine name which is aligned horizontally to the left side of the bottle price. By using the training set we were able to identify similar characteristics that are in the form of a table in each catalog. The accuracy improved when we tried to cluster prices based on their right and top coordinates on the page. To quantify the accuracy rate we compared the string output of our algorithm with the true string value for each feature in our test set. Indeed, overall accuracy improved over time to reach 72%. In addition, the accuracy for each feature is wine name: 61%, bottle price: 81%, case price: 73%. In addition, we identified different types of errors which might affect the accuracy of our algorithm.

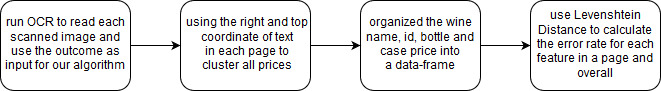
## **Introduction:**

Creating an archive of vintage wine catalogs can help wine economists, producers, and other interest groups to have access to this information for future analysis which might help them gain a better insight and understanding of the wine market over time². The Sherry Lehmann¹ catalogs contain numerous catalogs with different designs, outlines, formats and colors from the 30’s to 80’s. Each individual page or series of catalogs might have a unique design which contains only images, articles, actual wine catalogs, or even other alcoholic beverages. Extracting the information manually from the images might be easier but is time-consuming and error-prone. Therefore, we needed to develop an automated system to extract the relevant information with high accuracy from the scanned wine catalogs.

We developed an algorithm to extract specific information (e.g, wine name, bottle price, case price) and organize them into a dataframe as an output. Furthermore, the code that we have developed for this type of data extraction can be useful for others who are interested in text extraction using Tesseract outputs. Tesseract is one of the most common open source OCR softwares available in the market (4). Furthermore, we anticipate that the interest groups for our project would be people who want to revise and improve our code as well as people who want to use the outputs from our code. Based on our available time and resources, we developed an algorithm for our approach. Therefore, by looking at different factors; for instance, time and accuracy, we decided to use Tesseract as a software to interpret scanned images. Because of the complexity of our problem, we needed to break down our approach step by step such as extracting all of the prices from the Tesseract output, then extracting the wine names and matching them with their respective prices. Finally, calculate the accuracy rate. For example, we cluster all of the prices first and then label them, instead of finding the label ‘bottle’ in the page and then extract the prices. Even if we missed the label ‘bottle’ for any reason (Tesseract error), we can still extract prices and wine names.

## **Methods:**

Our data is in the form of high quality jpeg files (independent variable) which are not suitable for data analysis. A common method for reading and extracting text out of an image is called Optical Character Recognition (OCR). The three OCR software methods we considered are Tesseract, PdfPenPro, and ABBYY Reader. Among those three only Tesseract and ABBYY Reader return the coordinates of the strings within a page. Indeed, using the coordinates within the page to extract features is important for us because our algorithm works based on the position of the text within a page. Moreover, by manually comparing the output of Tesseract and ABBYY Reader we decided to choose Tesseract because it satisfied the needs of our project.



**Figure 1**: Flowchart of our approach to extract features from wine catalogs, see appendix three for more in depth flowchart

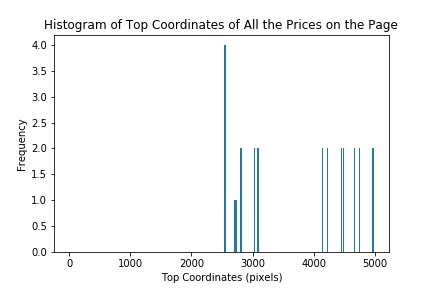
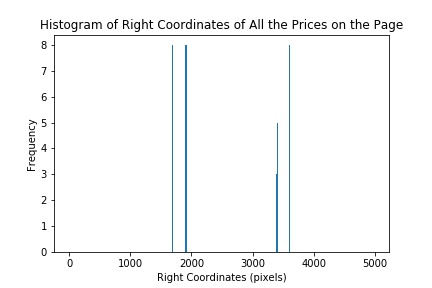
Running each image through Tesseract allows us to obtain all of the text as a first step. If Tesseract does not return any text we can assume there is a high chance that this page does not contain any relevant information. Thus, our algorithm is detecting pages which do not contain any wine prices. However, if Tesseract returned text then we used the pixel coordinates (left, right, height top, bottom) of every piece of text in the scanned images which tells us its location on the page. Indeed, we used Tesseract output text and coordinates to construct the rough outline of each page, for example, how many bottle price columns we have in a page (Figure 1).

For our supervised machine learning algorithm, we needed to create a labeled training set. For that we used three pages that were manually labeled with the information they contain. In fact, a very small sample size to test and train our algorithm is one of our major issues which we had in terms of training and quantifying our result. To solve this problem we chose catalogs which have similar outlines to narrow down our target catalogs in the main collection. We created a truth table for both the training set and test set manually which contains all of the correct values for our chosen features (wine name, id, bottle and case price).

We chose pages that have at least one column of bottle/case price which contains a hidden tabular format where the product name is to the left of the prices. The catalogs, which are mostly black and white in color, or have few colors, are also highly important in our selection because the Tesseract output accuracy dropped when the catalog is multicolored (specifically light colors and background).

The bottle price is the most important feature to extract because we used it as the first step in our algorithm. Also, it is an indicator if a scanned image is a wine catalog or not. So, as a first step we used a regular expression to get any text which is likely to be a price from all of the text that Tesseract returned.

Based on manually looking at different catalogs, all prices were right aligned (Figure 2). Therefore, we clustered based on the right coordinates because we wanted to build boundaries and decrease the variance of fluctuation between prices. One problem we faced was the right coordinate of prices in the same column didn’t have similar values because the scanned document was either rotated during scanning or for some other reason. The tilted page also affects Tesseracts accuracy as we came to realize, so we added a method to our program to rotate the page based on how much the price column was tilted. As a result, we fit a model based on the right coordinates of the prices using linear regression to calculate the degree of rotation. (Appendix 2). Then, after we rotated the page we ran the image through Tesseract again and repeated the steps previously mentioned.



**Figure 2**:Histogram of the right and top coordinates of prices in the wine catalog number 69.

Each wine might have both bottle and case price so as a first step we clustered the prices based on their right values and paired those which are closer to each other. Using this method also helped us to distinguish the number of columns in each page by looking at the distribution of each price cluster within each page (Figure 2). After finding the clusters based on the right coordinates (clustering the columns or the two halves of the page), we used the top values of each cluster to determine the boundaries of the rows. This method helped us recognize whether there is a bottle and case price or only one price in the column and label them accordingly.

The last feature to extract is the wine name, which is to the left side of the bottle price as we observed in our training set. We tried to estimate the margin of error based on the bottle price height and compare it with the wine name height in the training set manually. Also, the wine name is usually aligned with the bottle price horizontally.

Although we can detect the wine price, when there are two lines for a wine name, we cannot correctly extract the name. Therefore, we designed our algorithm based on selecting the left coordinate of each bottle price, then extract all of the text which might include the number ID and extraneous text.

The number ID (3 or 4 digits) is usually to the left of the wine names and has either the same or smaller font size and appears to be unique for each page. Indeed, the number ID should have been saved in a different column, but our algorithm did not separate them because of the limitation of our algorithm. Moreover, if there is any text in the margin of the page on the left side, our algorithm might pick them up and include those text as part of the wine name which is a problem we have not yet addressed in our algorithm. Those factors could have affected the calculation of our error rate. Eventually, we combined the wine name (plus the number ID), bottle and case price and created a dataframe as an output of our algorithm to calculate the error rate as the last step.

Calculation of the error rate is important to be able to validate the accuracy of our algorithm. Indeed, every time we change the variables in our algorithm it affects the final output so we can improve our algorithm and adjust different variable thresholds over time. We calculated the error rate on the testing set based on comparing the truth value (from the test set) with our algorithm’s output.

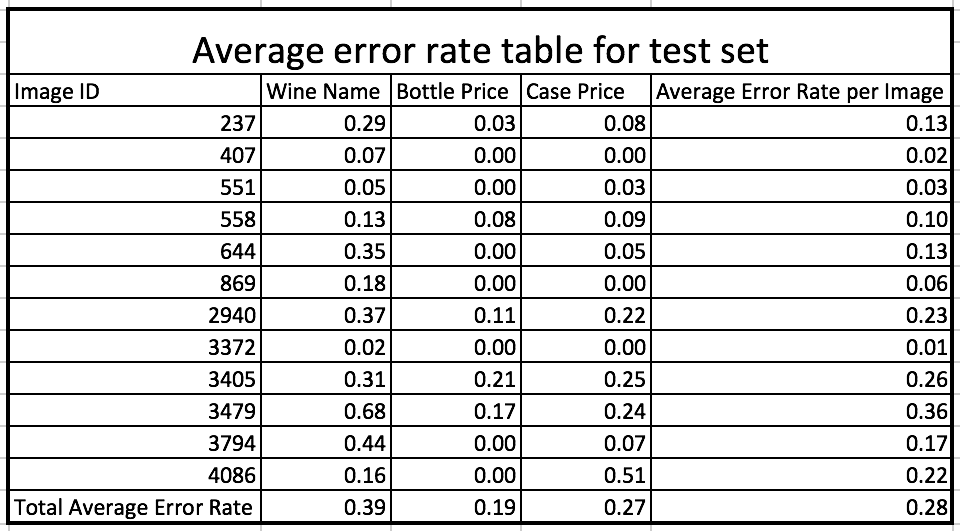
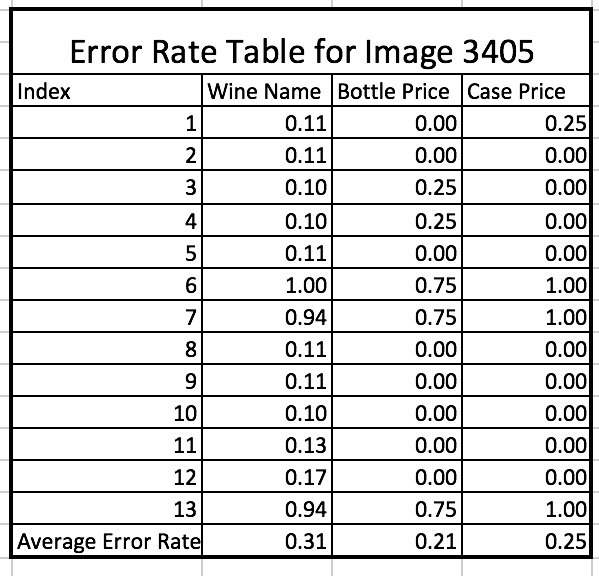
We have used the Levenshtein Distance (LD)³ which measures the similarity between the string in our truth table (test set) and the string in our algorithm’s output table. The method calculated the cost of deletion, substitution and insertion of characters and returned an integer. The integer is a positive number from zero to the length of the longest string of the two strings (zero means both strings are identical). If the LD is greater than the character size of the truth table, then the error rate is 1 (100%). To calculate the average error rate for that specific cell we divide the LD by the number of characters in the corresponding cell in the truth table. For instance, the name of the wine in the fifth row. Moreover, we calculated the average error rate for each feature in the page, and we average those error rates (for each page) to calculate the overall average error rate for all features and total average error rate (Figure 3).

Then, we can predict that it will most likely work well on other images. One critical point is that the order of the bottle price between the truth table and our program’s output should be the same because we compared each row one by one and their corresponding elements such as the wine name, bottle and case prices.

## **Results:**

To calculate our algorithms performance, it is important to quantify the results that give us a more in-depth description of our algorithm’s success. Recognizing different types of errors which directly affects the accuracy of our output is one of our main challenges which we have have tried to detect and address over time. Some examples of the type of errors are: Tesseract reads the price but separates the digits, returns the wrong digits, or totally misses the values.

As we mentioned, the Levenshtein Distance algorithm calculates and weighs the cost of deletion, substitution and insertion character and returns the final result as an integer. By quantifying our findings, (although it’s not the most comprehensive way) we were able to measure the error rate for each of the features as well as the total average error rate (Figure 3).

**Figure 3**: (on the left) Average error rate table for each feature of each image and overall average error rate for each feature in test set, and overall error rate in the test set. (on the right) error rate for each feature entry in our table and average error rate of each cell of image 3405.

The table above reports the error rates for each image in our training set. The three types of error that we looked at by column are the following: wine name, bottle price and case price as well as the total error rate for each individual page. The three types of errors gave us a better idea of how well our algorithm is performing for each specific feature we are trying to capture.

The table in figure 3 shows that the column with the lowest error rate is the bottle price column with an error rate of 19%. The column with the highest error rate is the wine name column with an error rate of 39%. The wine name error rate is higher because our algorithm extracts other pieces of text such as number IDs, and wine descriptions when trying to capture the wine name. Furthermore, when bottle price error rate increases, the wine name error rate increases simultaneously because wine name extraction depends on wine bottle price. Overall, our algorithm’s total error rate is 28% (Figure 3).

## **Discussion:**

Using supervised machine learning and statistical methods helped us develop an algorithm for the successful extraction of historic prices from Sherry Lehmann wine catalogs (wine name, id, bottle price, case price). As previously mentioned, each catalog has a different design and unique characteristics. Therefore, creating a unique approach to develop an algorithm is a very challenging process.

Our approach was to identify and group those catalogs which have similar characteristics to create a training set which we labeled manually. Then, we developed our algorithm based on some facts we observed from the scanned catalogs we picked.

As a fundamental step, we extracted all of the prices from the Tesseract output and tried to cluster them into at least one cluster based on their right coordinates. Then, within each cluster, we separated each row based on their top values. Next, each row has one bottle price which might pair with a case price or not. Afterwards, we use the bottle price and match each one of them with their respective wine name which is usually in the same line as the prices. Finally, we saved the wine names, bottle and case prices in a dataframe.

Furthermore, we calculated the accuracy rate using the Levenshtein Distance (LD)³ for each feature within a scanned page. In addition, we calculated the average error rate of the whole training set for each feature as well as the total error rate. For instance, when we compare the name of the wine, the LD calculated deletion, substitution and insertion based on what we set as a penalty number and returns the total error rate. Therefore, by adjusting different variables in our algorithm we try to predict how well our code will perform based on the test set.

The LD method returned an integer for each cell which we then divided by the number of characters in the cell to calculate the percentage of error. In fact, by subtracting one from the error rate we can get the accuracy rate for each feature and the overall accuracy for each catalog. Indeed, over time we have improved our algorithm to reach 72% overall accuracy when we tested our algorithm using the test set, and the accuracy for each feature is: wine name 61%, bottle price 81%, case price 73%. This result is only based on our training set where pages contain similar outlines or characteristics. We can’t predict what our error rate will be if we run our algorithm on all 4111 scanned catalogs.

We have recognized two types of problems which decreases the accuracy rate of our algorithm. First, the Tesseract software errors which we label as preprocessing errors; second, those errors which depend on our developed algorithm are post-processing errors. To improve our overall error rate, we would have to address those two type of errors in our algorithm by preprocessing and post-processing.

In preprocessing, we faced problems such as Tesseract missing text completely or partially (incorrect format) because of page coloration, rotation, font and design. For example, the scanned catalog can be brightly colored, slightly rotated when scanned and have an unfamiliar font that Tesseract could not detect. We were able to solve rotation by using linear regression to find the best fitted line using the right coordinates of the prices (Appendix 2).   
 In post-processing, we encountered issues with extracting features correctly, labeling prices incorrectly and fixing prices from partially incorrect Tesseract outputs. For example, when Tesseract separates the price 41.15 into 41 and 0.15 or when Tesseract does not capture bottle price, our algorithm puts the case price as the bottle price. We were able to fix some of these prices by using regular expressions to recognize these patterns and replacing them with the correct price.

# 

# **Future Direction:**

We are aware that our algorithm has limitations and because of that there are still multiple ways that we can improve our algorithm to achieve a higher accuracy. These problems range from extracting the bottle price with higher accuracy from the Tesseract output and labeling them, and decreasing high wine name error rates which is directly correlated to the accuracy of bottle price extraction. Therefore, the solution is to separate wine number ID from wine name and save it in a separate column and to add a method to capture those wine names which are in a two line format.

Furthermore, we need to extract and build the price list extraction better even if the Tesseract format was not accurate. For instance, if Tesseract separates a price into two parts which have commas and/or decimal our algorithm couldn’t recognize that. Statistically speaking we need to create a larger training and test set which will help us train our algorithm better and calculate a more representative error rate. Furthermore, we can use both ABBYY Reader and Tesseract output to develop a method to cross check the text each one provides and combine into one piece text to increase the confidence level of correctly extracting the text from the catalogs.

**Conclusion:**

Achieving high extracting accuracy rate from scanned wine catalogs using automated feature extraction requires different approaches. Our algorithm is just one approach that extracts specific features from wine scanned catalogs which have a similar outline. Although we can improve our algorithms accuracy rate over time, we have to consider our approach limitation and those factors which our algorithm will not be able to solve. Moreover, the extracted data might be used by those who are interested in further research and analysis to recognize how different factors affect wine prices¹. In addition, the code we have developed might be used by people who are interested in extracting data from any catalogs which have similar types of outlines.

# 

# **Appendix:**

### 1. Learning Outcomes:

Working as a group to complete the ten week project gave us a unique opportunity to recognize how important teamwork, communication and responsibilities are. Understanding and writing the proposal for the timeline is critical for team success although what we have learned is, how to change our approach and adjusting our milestones while moving forward each week. In addition, we improved our ability to communicate with each other, to explain and convince when needed.

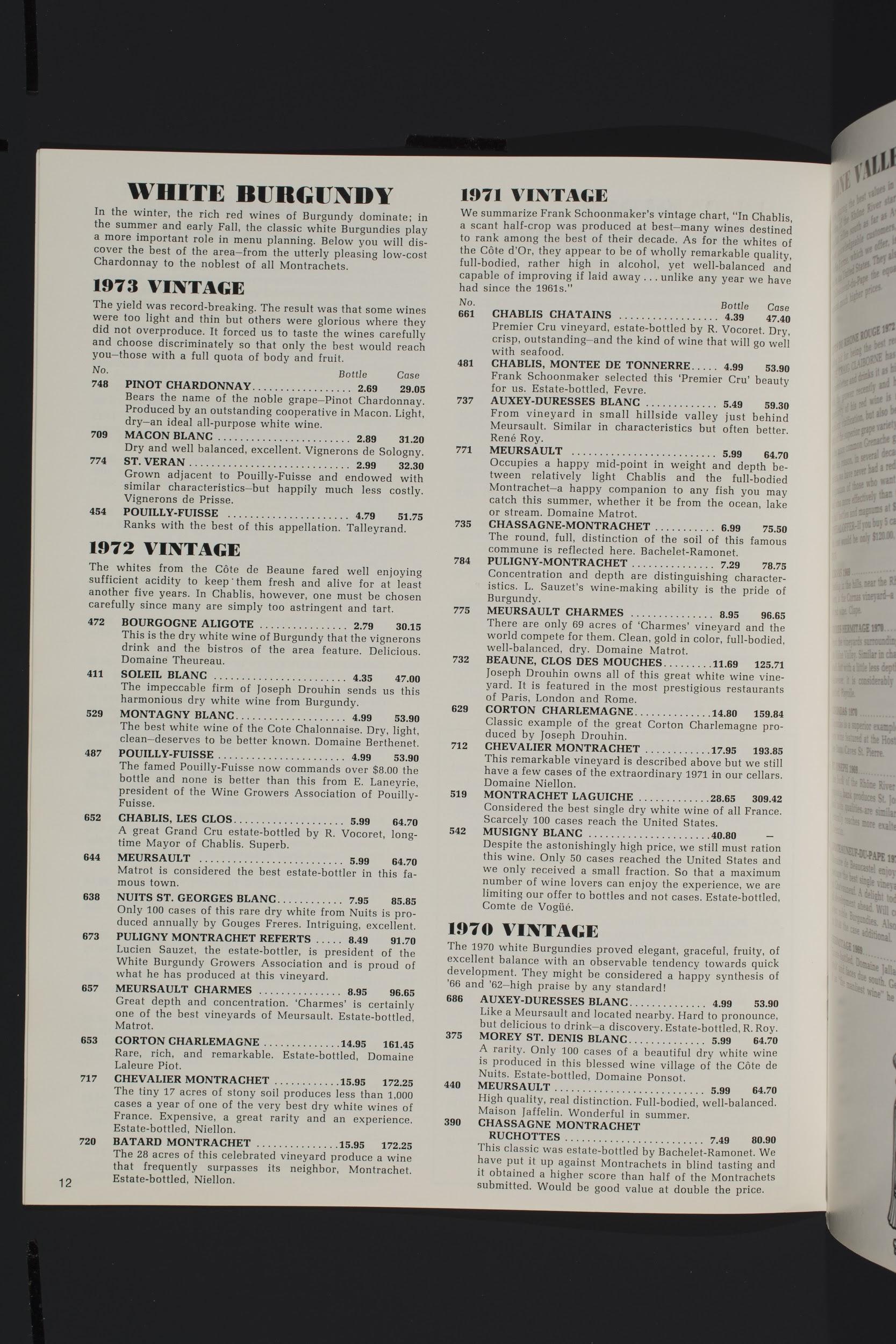
Time management individually and as a group to meet, discuss and work together is another critical lesson we learned. We have learned although we have a ultimate goal approaches and methods we have used. Being exposed to new methods and trying to learn new technologies and techniques efficiently help us to improve how we approach new problems. Data extraction and preparation is an important part of data analysis and we learned over the course of this project that a significant amount of time is dedicated to preparing specific data we are interested in analyzing.

Overall, this group project experience has helped us become more well rounded in every aspect of being a successful worker. It gave us the tools of learning how to learn, discovering new knowledge, and most importantly working efficiently and professionally with a group.

### 

### 2. Page Rotation Procedure:

We have used linear regression to calculate the error of each right value of bottle price and then based on that calculate the angle of regression line. We chose the top value as the independent variable and the right coordinate as a the dependent variable in our linear regression model. And based on the slope of the regression line we can find the angle and rotate the page respectively if the angle of rotation is more than 0.5 degree (our threshold). Also, when calculating the slope we consider multiple clusters of wine bottle price which have 4 values or more if there are more than one price cluster exist in the page. Because, more data point smooth the linear regression and decrease variance.



(on the left) Original scanned catalog number 551. As you can observe, this page was tilted so our algorithm rotated this page, then run the OCR again and extracted the features

### 3. Flowchart:

### 

### 4. Tesseract:

Tesseract is an [optical character recognition](https://en.wikipedia.org/wiki/Optical_character_recognition) (OCR) engine for various operating systems. It is [a free software](https://en.wikipedia.org/wiki/Free_software). In 2006 Tesseract was considered one of the most accurate open-source OCR engines available. Tesseract development has been sponsored by [Google](https://en.wikipedia.org/wiki/Google) since 2006. Tesseract was originally developed at Hewlett-Packard Laboratories Bristol and at Hewlett-Packard Co, Greeley Colorado between 1985 and 1994.

<https://en.wikipedia.org/wiki/Tesseract_(software)>

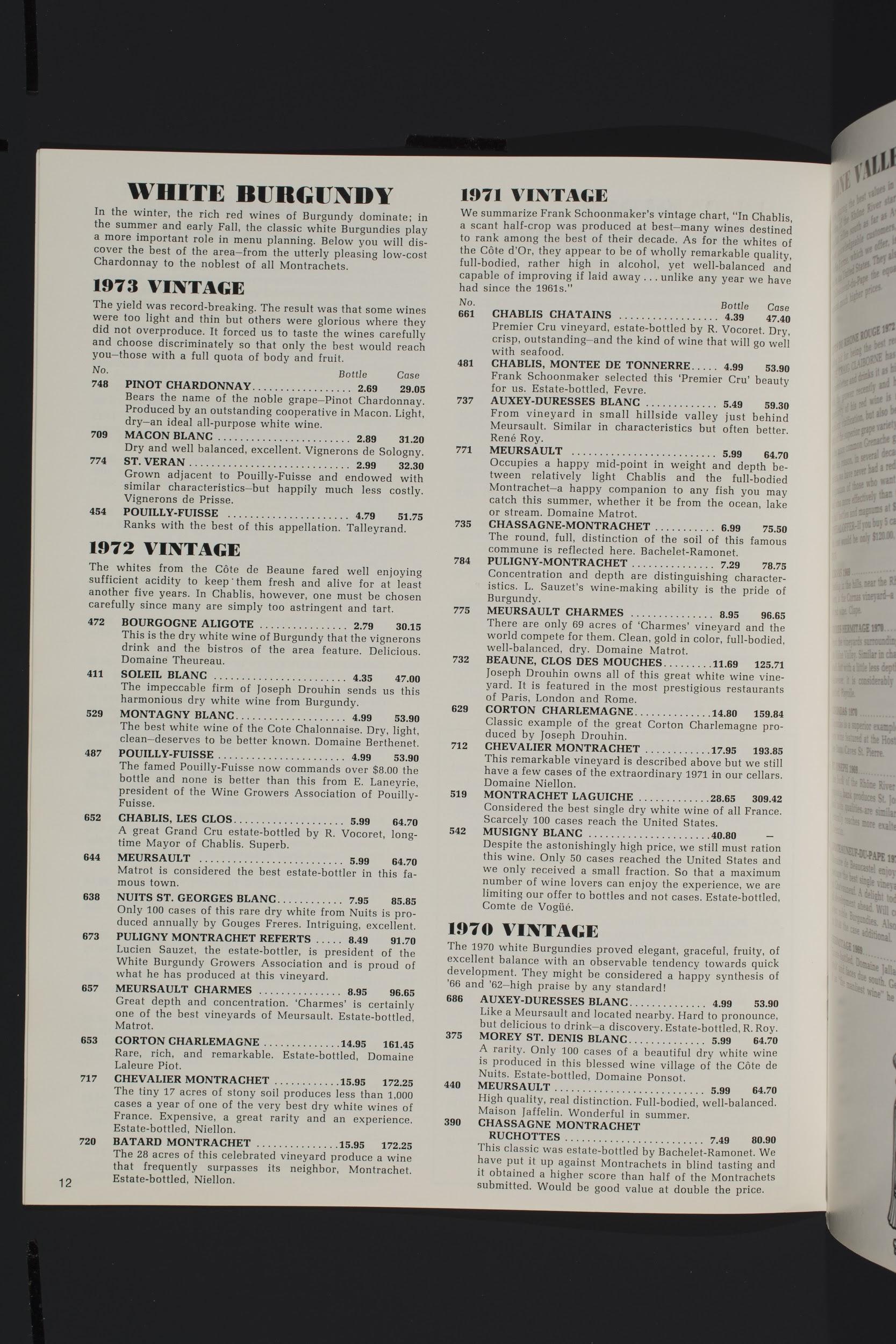
### 

### 5. Output vs. Truth Table:

### The truth table was created manually (right) vs our algorithm output (left) for scanned catalog number 551 in our training set (overall error rate for this page is 2.6%, wine name 4.8%, bottle price 0%, case price 3.1%).

(img\_ID: image id, name: Wine Name and wine ID)



.

# 

### 5. Software Versions:

* Python 3.6.4
* Mac OS Sierra Version 10.12.6
* Tesseract version 3.05.01
* Python Packages and Versions:

1. pytesseract 0.2.0
2. Pillow 5.0.0
3. Imageio 2.3.0
4. pandas 0.22.0
5. numpy 1.14.0
6. matplotlib 2.1.2
7. Scipy 1.0.0

# 

# **Bibliography:**

1. “Price the Vintage (Alpha2).” *Price the Vintage (Alpha2)*, Dec. 2017, <https://ptv.library.ucdavis.edu/>
2. Mayyasi, Alex. “The Price of Wine.” *Priceonomics*, 29 Mar. 2013, <https://priceonomics.com/post/46618070248/the-price-of-wine>
   1. (this is article about wine economy and business)
3. Gilleland, Michael,“Levenshtein Distance, in Three Flavors.”, <https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm>
4. Tesseract <https://github.com/tesseract-ocr/tesseract>
5. Importance of Feature Engineering <https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/>