Python for Wine-Lovers: 2017 Wine Enthusiast Review Analysis

Claire Boetticher and Tina Phu // MSDS_430, Fall 2018

Project description

Our project team ingested and analyzed a Kaggle dataset of 130,000 wine reviews scraped from Wine Enthusiast magazine during the week of November 22, 2017 (reference). This collection held obvious appeal as we both enjoy wine, but more importantly, offers valuable opportunities for data exploration, statistical description, quantitative analysis, text analysis, and visualizations. The reviews cover a broad range of countries of origin of wine, varietals, regions, and vintage years, which made for meaningful exploration and analysis.

The original dataset contains a combination of 13 numeric and text fields:

Field	Description	Unique Values
Country	Country the wine is from	43
Description	A few sentences from a sommelier describing the wine's taste, smell, look, feel, etc.	119955
Designation	The vineyard within the winery where the grapes that made the wine are from	37,976
Points	The number of points Wine Enthusiast rated the wine on a scale of 1-100 (though they say they only post reviews for wines that score >=80)	NaN
Price	The cost for a bottle of the wine	NaN
Province	The province or state that the wine is from	425
Region_1	The wine growing area in a province or state (e.g., Napa)	1,229
Region_2	Sometimes there are more specific regions specified within a wine growing area (e.g., Rutherford inside the Napa Valley), but this value can sometimes be blank	17
Taster_name	Name of the person who tasted and reviewed the wine	19
Taster_twitter_handle	Twitter handle for the person who tasted and reviewed the wine	15
Title	The title of the wine review, which often contains the vintage, which can be extracted as year	118,840
Variety:	The type of grapes used to make the wine (e.g., Pinot Noir)	707
Winery	The winery that made the wine	16,757

Data preparation

To prepare the data, we loaded the CSV file from Kaggle into a DataFrame first. We then extracted the 4-digit year of the wine from the “Title” field to create a “Year” field for analysis over time of vintage, converting from string to date-time format.

However, extracting the wine year from Title column was not as straightforward as expected.

- Some wines had two or even three sets of four-digit numbers in its name (for example, “Foxen 7200 2012 7200 Grassini Family Vineyard Cabernet Sauvignon”).
- A few wines had no year but a four-digit number in its name that we verified was not its year (for example, “Hush Heath Estate NV Balfour 1503 Classic Cuvée Sparkling”).

To solve this problem and extract the correct year from the name, we initialized an empty list (*yearlist*) and used a for-loop to iterate through each wine title to extract the correct year from the title and append the year to the list. The loop contains the following:

- A call to the user-defined function *NumberFromStrings* to extract all digit values from the name into an empty list
- A list comprehension to extract only 4-digit characters from the resulting list
- If-else conditional statements to check for length of the list, compare values within the list, and append the proper year to *yearlist*.
    - If length of list is equal to one and the single item in the list is greater than 1800 (we know that the wines in the list are no older than 1800), then we set this as the year.
    - We know that the year must be between 1800 and 2020. If length of list is greater than one (the lists were no longer than three, that is, one single wine name has no more than three four-digit characters), then we compare value of items and set the year as the larger of the values, unless the digit is greater than 2020.
    - All wines with no year at all were assigned “No year”.

Once these steps were complete, we appended the results of *yearlist* to the dataframe as a new column, *Year0*.

However, the years listed under *Year0* are in string format. To properly convert these values to date format while accounting for “No year” values, we performed the following:

- Initialized *new_year* as an empty list
- Defined the function *str_to_year* that uses the Pandas function *_to_datetime* to convert a string to date format and extracts the year from the date.
- Used a for-loop to iterate through the *Year0* column and an if-else conditional statement that replaces “No year” values with null using the NumPy *null* function, and if the value is a four-digit string, call the *str_to_year* function on the string to convert it to date format. The resulting value is then appended to *new_year*.
- Set results of *new_year* as new column *Year* in the dataframe.

Results of data duplicate removal are as follows:

Shape before: (129971, 15)
Shape after: (119988, 15) - 9983 duplicates removed

Text analysis

The Description field provided varied and descriptive text that offered an opportunity to practice pre-processing and data cleanup, and also a means to conduct basic text analysis and natural language processing techniques. Pre-processing steps included:

- Converting all text to lower-case
- Removing punctuation, except hyphens so that compound adjectives (e.g., oak-driven) remain as one word
- Removing stopwords using NLTK
- Creating a series for most-frequently and least-frequently occurring words for exploration purposes
- Tokenizing using NLTK to create semantic units for processing (reference)

Bigram analysis

N-grams of texts are extensively used in text mining and natural language processing tasks, described as a set of co-occurring words within a given window (reference). Once the "Description" field was tokenized, we conducted bigram analysis to explore the most "meaningful" two-word combinations from that field.

Sentiment analysis

We also conducted sentiment analysis to explore tasters' impressions numerically using the VADER (Valence Aware Dictionary and sEntiment Reasoner) Sentiment Analyzer from NLTK (reference). This sentiment analysis tool was created for working with messy social media data and provided a quick means for evaluating the tone of wine descriptions from the various tasters. From the NLTK DataFrame created on Description, we used the 'compound' score as the "definitive" rating of the sentiment for simplicity, renamed as 'Sentiment,' while the other three columns are a more detailed view of negativity, neutrality and positivity of the review. These provided a means for analyzing descriptor text trends across the dataset, including correlation with year of vintage, points, and price.

Results

Table 1: End Result Summary

Descriptive statistics that summarize the central tendency, dispersion, and shape of the dataset's distribution.
Pulled using describe() function

	Points	Price	Year	Description_count	Avg_description	Sentiment	neg	neu	pos
count	120072.000000	111663.000000	120072.000000	120072.000000	120072.000000	111472.000000	111472.000000	111472.000000	111472.000000
mean	88.442068	35.621513	1939.049162	40.421597	5.048896	0.521840	0.021729	0.779034	0.199235
std	3.092782	42.095332	372.413056	11.203481	0.398798	0.364969	0.046143	0.130331	0.131415
min	80.000000	4.000000	0.000000	3.000000	3.703704	-0.928800	0.000000	0.118000	0.000000
25%	86.000000	17.000000	2008.000000	33.000000	4.770833	0.318200	0.000000	0.689000	0.103000
50%	88.000000	25.000000	2011.000000	40.000000	5.027027	0.636900	0.000000	0.782000	0.194000
75%	91.000000	42.000000	2013.000000	47.000000	5.303030	0.817600	0.000000	0.874000	0.291000
max	100.000000	3300.000000	2017.000000	135.000000	8.200000	0.989200	0.675000	1.000000	0.800000

Figure 1: Number of Reviews by Year

These reviews primarily cover vintages from 2010-2014, dropping off significantly afterward. The reviews were scraped in 2017, explaining why no year-old 2018 vintages are covered.

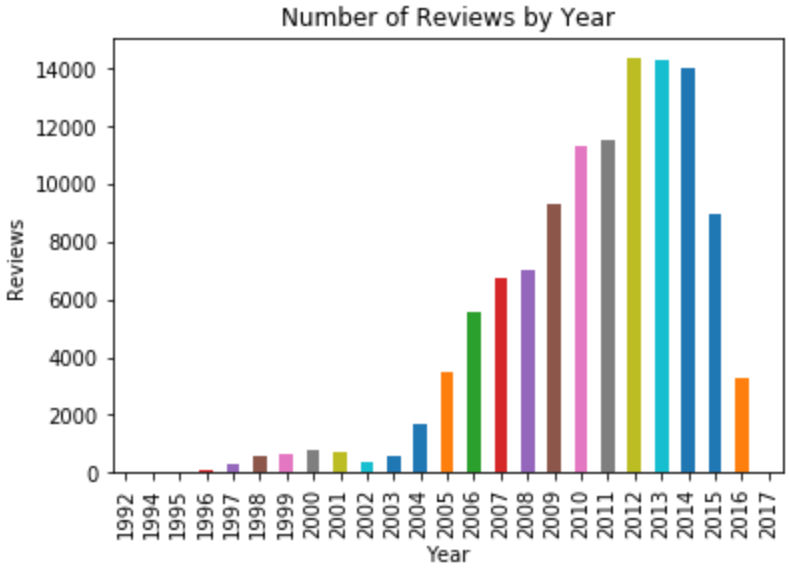


Figure 2: Top Description Words by Occurrence

Wine is the obvious result for most frequently-occurring word since that is the data's focus. It makes sense, as well, that reviews would include categorical wine description terminology such as "flavors," "aromas," "nose," and "palate."

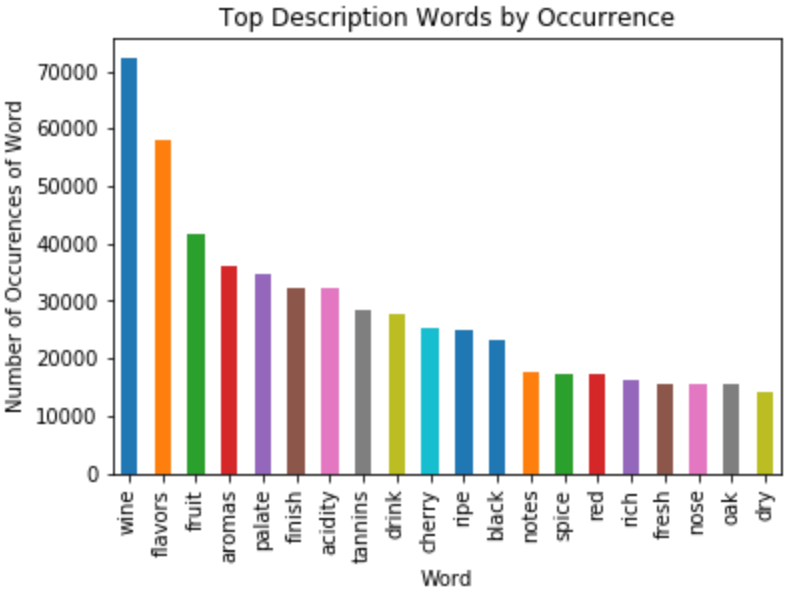


Figure 3: Top Bigrams by Occurrence
Flavor combinations (“black cherry”) and varietals (“cabernet sauvignon”) dominate this bigram frequency chart, as expected. We also see some common tasting terminology, such as “nose, palate.”

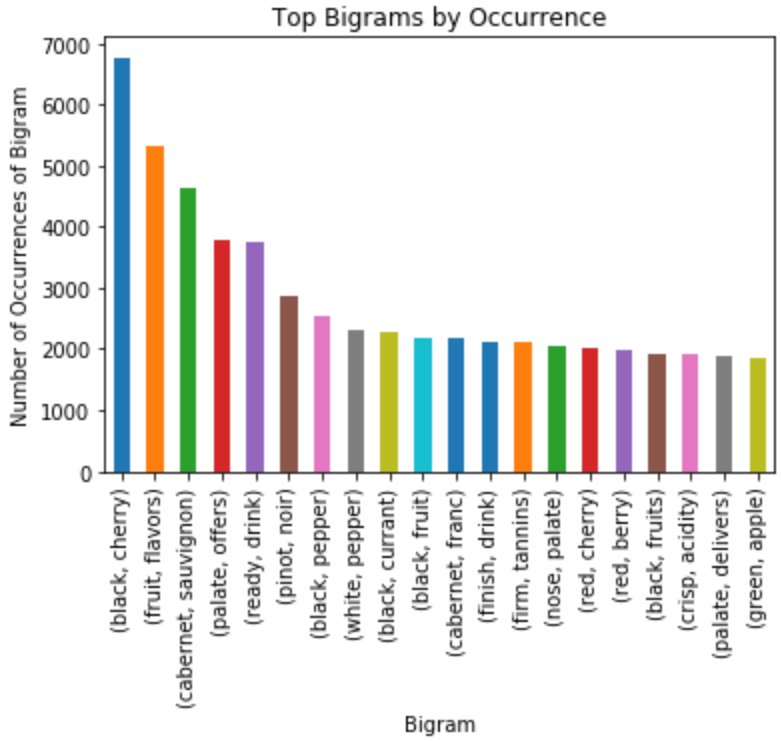


Figure 4: Sentiment Scores by Year
The sentiment scores for wines run the entire spectrum of -1 to 1 for most years. In the older vintages before 2000, the scores tend to be higher. It’s possible that some of the wines reviewed in that age range were of an exceptional vintage (reference). If a reviewer went into a tasting knowing they were experiencing that situation, it’s likely they would be predisposed to reviewing positively.

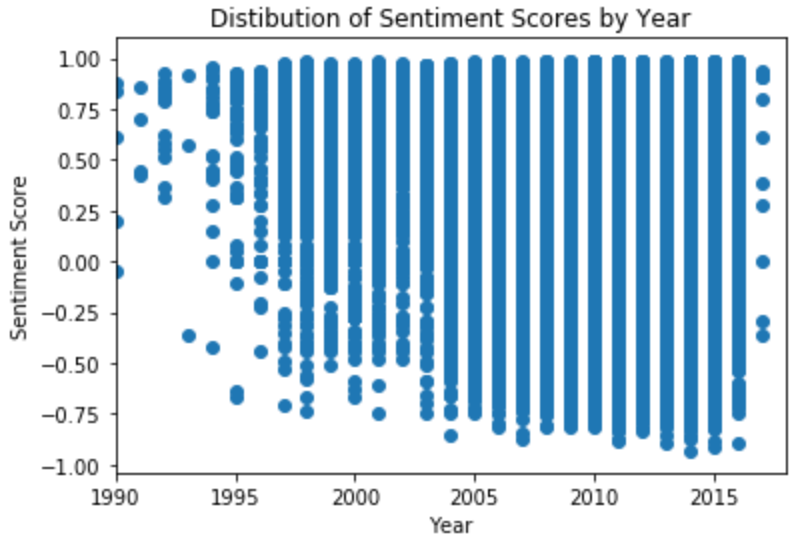


Figure 5: Price versus Sentiment Score

Most wines in this dataset cost below $500. This concentration makes it challenging to make conclusions about the correlation of price and sentiment score, but it seems that there are slightly more positive sentiments associated with more expensive wines up to that point. The outliers in the very-high prices range all tend to have much higher sentiment scores, so perhaps those wines are rare and/or of an exceptional vintage.

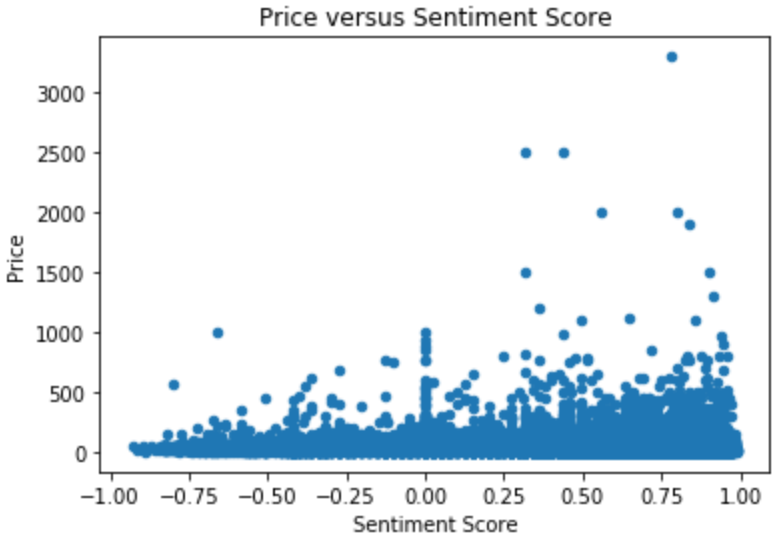
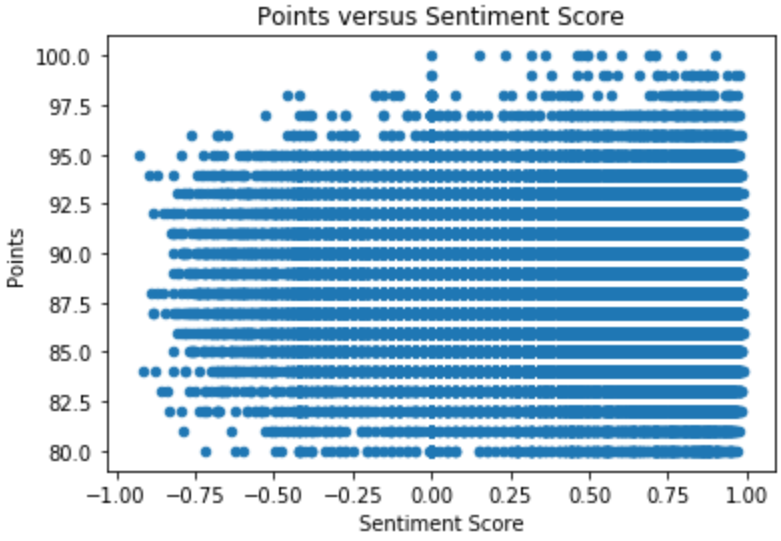


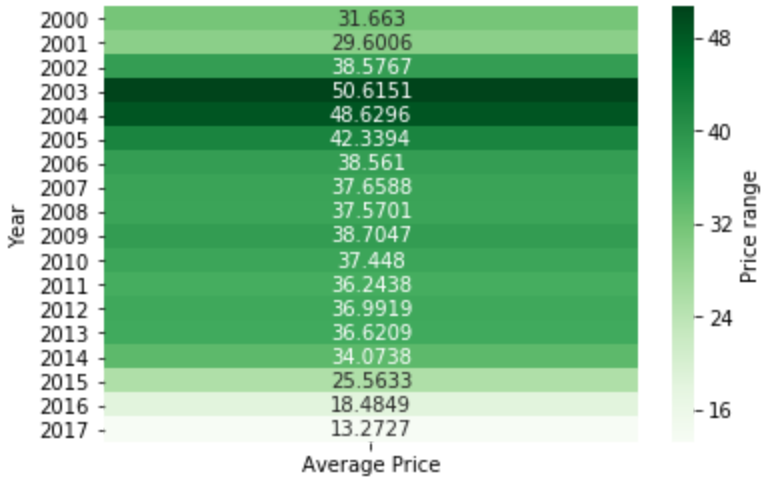
Figure 6: Points versus Sentiment Score

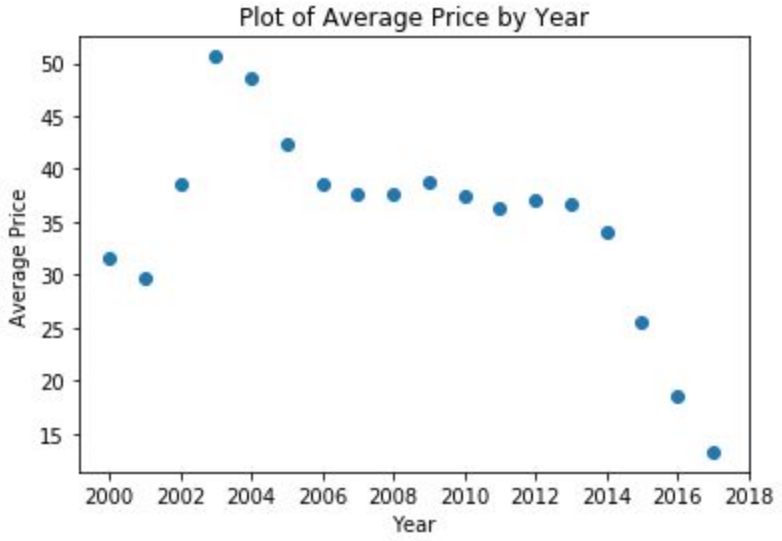
Sentiment scores appear distributed evenly across all points assignments. For the highest point assignments (97.5-100), there are no negative scores, only neutral to positive.



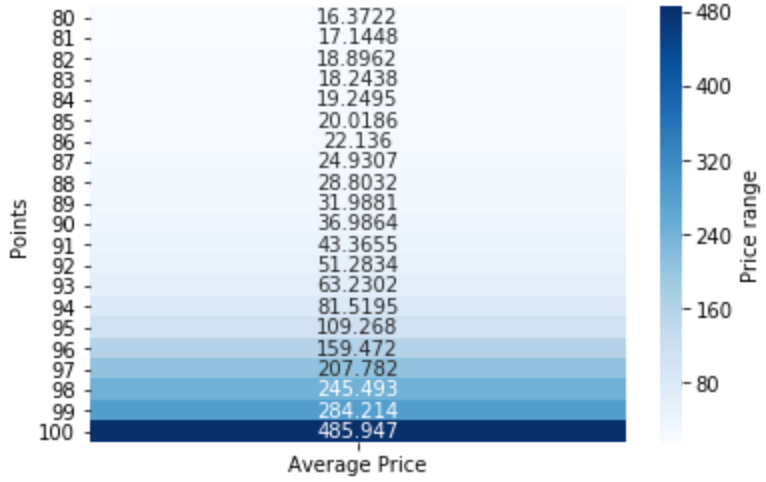
Figures 7 and 8: Average Price by Year Since 2000 (Table and Scatter Plot)

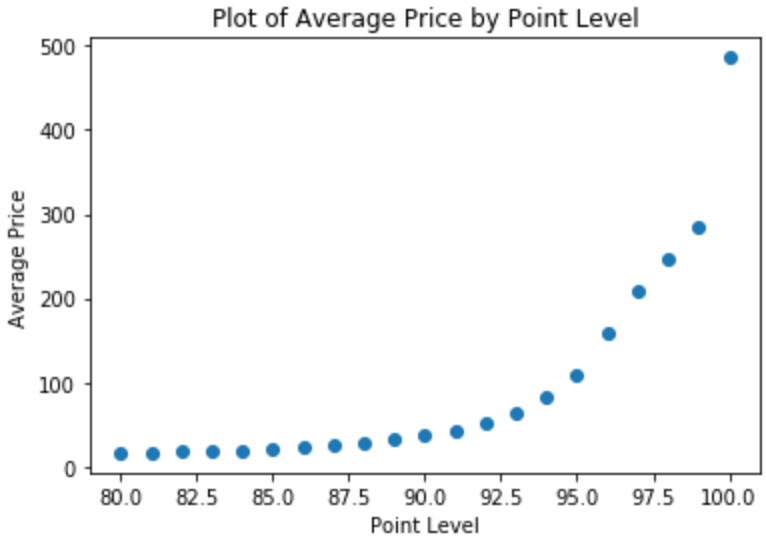
We chose 2000 since it is the year we start to see wider distribution in prices in scatter plot. The 2003 vintage year saw the highest average prices (~$50), with the following two years in second and third place in terms of highest average price. Older wines tend to have a much higher price in this dataset. There are much fewer wine reviews for vintages older than 2000, so those higher-priced wines greater impact on the average price as well.





Figures 9 and 10: Average Price by Point Level (Table and Scatter Plot)
As positive reception of a wine increases, average price does too, though average price seems to rise sharply between 99 and 100 points.





Summary and ideas for next steps

This combination of numeric and text data, with a year field that had to be extracted and custom-formatted, offered a great opportunity for exercising many of the concepts we have covered over the course of the quarter: string formatting, conditional statements, calculations, user-defined functions applied to DataFrame columns, and exploration via pandas, matplotlib, and seaborn.

Some interesting next analyses, given more experience with language modeling and machine learning, could include predicting classifications of wine (by variety or region, for example) or the development of a recommendation engine based on reviews as a training set. It would also be an interesting experiment to scrape Twitter data from the reviewers, since handles were given, to compare formal reviews in a publication like Wine Enthusiast versus more casual social media posts about the specific wines under review.

References and requirements

NumPy: to create array of dimensions to use for scatter plots
Matplotlib: used for all visualizations
Pandas: used for Dataframe creation, exploration, function application, and custom analyses and calculations
External data from a file: CSV file containing 130,000 wine reviews from Kaggle (link to original dataset)
Calculations and conditional formatting throughout notebook
User-defined functions:

- *NumberFromStrings* - extracts all digits from a string into a list
- *Str_to_year* - converts a four-digit string into date format
- *Avg_word* - average word length for description field
- *stopwordRemove* - apply a custom stopword list beyond NLTK's default
- *Find_bigrams* - find word pairs for analysis
- *Nltk_sentiment* - calculate sentiment score using NLTK VADER

Modules imported and usage:

- Pandas: data preparation and exploration
- NumPy: calculations and arrays for visualizations
- Regular expressions: formatting with RegEx
- Math: rounding
- String: string formatting
- Matplotlib, Pyplot, Ticker from Matplotlib: visualizations
- NLTK: text analysis (bigrams and sentiment)
- Itertools: iteration over the DataFrame
- Collections: creation of the bigram counter
- Seaborn: color coding for average price by point and by year tables