1 Introduction [0.5 pages]

Wine is both an everyday and luxury good with both subjective and quantitative properties. Many aspects of wine can be described as a matter of fact (e.g., white or red, sparkling or not, origin). But wine reviewers and sommeliers find employment due to the fact that an astounding variety of wines exist and wine preferences, of course, depend on one's taste. So if one was to enjoy a red wine from California because it tasted like cherries, blindly recommending another California red wine would be useless if it were to taste like cedar. Since much of the wine recommendation systems are currently built around quantitative analyses, I am aiming to build an app which can recommend wines to users relying almost entirely on subjective descriptions. The app relies on a dataset of $\sim 120,000$ unique wine reviews scraped from Wine Enthusiast Magazine, a leading wine publication with respected critics writing about wines from all around the world. I use several NLP techniques to quantify the text data. I then use a K nearest neighbors algorithm to predict other similar wines in the dataset. I also use the kmeans algorithm to cluster the wines. This allows us an opportunity to extract the most meaningful words in each cluster i.e., find what distinguishes each cluster of wines from the others. This could give us meaningful and unexpected insights into why certain wines may taste or feel similar.

2 Related work [0.5 pages]

As mentioned before, there are multiple approaches to categorizing and recommending wine. A previous CS229 paper used the actual chemical qualities of wine [1] which is somewhat similar to this approach [2] by Ewelina Osowska. An approach I find especially intriguing is a text-based analysis by Roald Schuring [3]. I am aiming to use a combination of empirical data and a text-based approach to most accurately capture the nuance of the problem.

Additionally, Vivino, which markets itself as the "most downloaded wine app" [4] uses both quantitative data and a user's review to recommend wine to the user. However, several internet discussions raise some potential issues. [5] [6] Namely, these are that the reviews on Vivino are sourced entirely from users on the app, which is not necessarily conducive to meaningful or informed data. Although one could argue this makes the recommendation more democratic, it

[1] Modeling Wine Quality from Physicochemical Properties `http://cs229.stanford.edu/proj2019aut/data/assignment_308832_raw/25895690.pdf`

[2] Clustering of wines - flat, hierarchical and fuzzy algorithms `https://rstudio-pubs-static.s3.amazonaws.com/474170_bce84d98324947f2ba9e4416b6a21465.html`

[3] Wine Embeddings and a Wine Recommender `https://towardsdatascience.com/robosomm-chapter-3-wine-embeddings-and-a-wine-recommender-9fc678f1041e`

[4] `https://www.vivino.com/about`

[5] `https://www.reddit.com/r/wine/comments/iod3oe/what_is_the_deal_with_vivino_why_is_it_popular/`

[6] `https://www.reddit.com/r/wine/comments/4mml6q/thoughts_on_vivino/`

might also lack the insight of an expert opinion which my model incorporates. The next is that Vivino may have an economic incentive to recommend certain wines over others since their business model is at least partially reliant on selling wines to those using the app.

3 Dataset and Features [0.5 - 1 pages] (Application Projects Only)

The dataset I use includes over $120,000$ wine reviews from Wine Enthusiast. I found the dataset on Kaggle. [7] Each row contains the name of the wine, the review from the magazine's critic, the wine's origin, the grape varietal(s), and other information such as the price and more specific details on where the wine was produced. An example from the dataset is given below:

Title	Description
Nicosia 2013 Vulkà Bianco (Etna)	Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, ...

As shown, I have chosen to select solely the review (usually $\sim 3-4$ sentences) as the raw data for the model. This was a design choice taken for several reasons. First, this project is intended to explore the utility of using text-based data to provide insights on wine. Second, some of the other fields of data such as the specific wine-growing region or price were incomplete, and omitting the examples which did not include these fields may have unintentionally skewed the data. Every example in the dataset included the review.

To preprocess the data, I had an idea that I would want to extract tasting notes such as `blackberry`, `tannic` or `purple`. To try and isolate the most important and intuitive data from the texts, I removed all numbers and stopwords (sourcing the list of stopwords from the NLTK module). Then I used the porter stemmer algorithm from NLTK to normalize the text by removing the suffixes of words. This allows us to find common sentiments in closely related words such as `purplish` & `purply`. Consider an arbitrary review as follows:

```
A Southern-Rhone style blend of 50% Grenache, 47% Syrah and 3%
  Petite Sirah from a single vineyard on Mount Veeder.  It's
  ''Estate'' in that the vineyard is winery-owned, but the
winemaking happens elsewhere, so it's technically not.  Smell and
  taste deep, dark black and blueberries with allspice, black
 licorice, cola and smoked red meat back-ups.  Strong wine, with
 firm, fine tannins and a small explosion of flavor as it sits in
 your mouth.  Longtable waits almost five years after harvest to
 release The Gathering so it's ready.  It's sort of ready...it'll
                start peaking in 2020 is my guess.
```

[7] `https://www.kaggle.com/zynicide/wine-reviews`

After preprocessing and normalization we get the less readable but hopefully more useful:

```
southern rhone style blend grenache syrah petite sirah single
vineyard mount veeder estate vineyard winery owned winemaking
happens elsewhere technically smell taste deep dark black
blueberries allspice black licorice cola smoked red meat back ups
strong wine firm fine tannins small explosion flavor sits mouth
longtable waits almost five years harvest release gathering ready
sort ready start peaking guess
```

4 Methods [1 - 1.5 pages]

The first algorithm of note is **TF-IDF Weighting**. TF-IDF which is short for term frequency–inverse document frequency is a method for turning text data into vectors which can be used for our task. The algorithm gives terms that occur less often more importance in determining the nature of the text at hand by the formula

$$\text{tf-idf} = \text{tf} \cdot \text{idf} \tag{1}$$

where tf is the frequency of a term in a document and idf is a measure of how rare a term is over the corpus. So a high information term will occur a lot in a certain document but rarely overall. Next, I used the **k nearest neighbors** algorithm to find a certain wine's most similar wines in the dataset. This algorithm works by finding the distance between an example and the other data. Then it returns the k closest datapoints. In this case, since our features were tf-idf weights of text data, the nearest neighbors should have similar descriptions and characteristics. Finally, I used the $K-means$ algorithm to cluster the data and find similarities within clusters. As shown in class, the k means algorithm is given as:

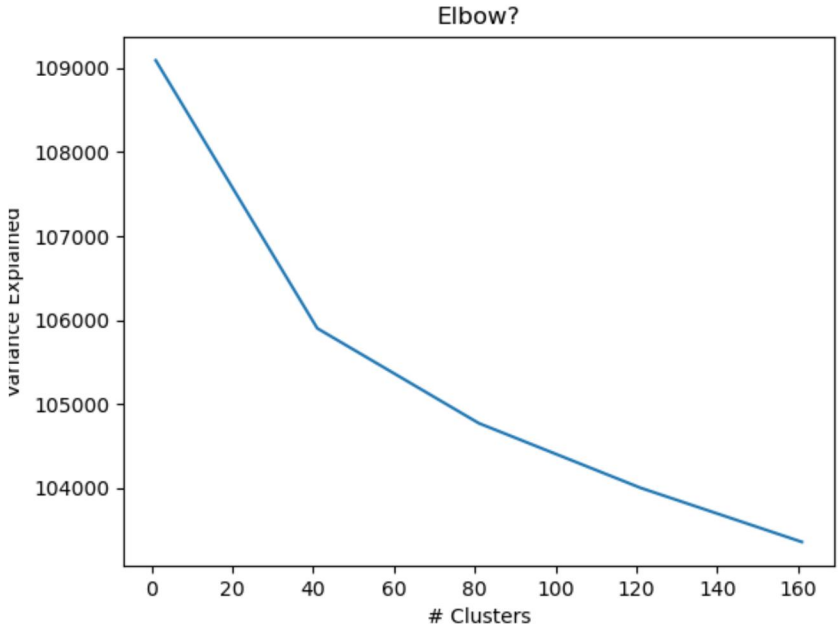
$$\begin{aligned} &1: \text{Initialize } k \text{ cluster centroids randomly} \\ &2: \text{Repeat until convergence:} \\ &\text{for every } i\text{: } c_i = \min_j ||x_i - \mu_j||^2 \\ &\text{for every } j\text{: } \mu_j = \frac{\sum_{i=1}^{n} 1\{c_i = j\} x_i}{\sum_{i=1}^{n} 1\{c_i = j\}} \end{aligned}$$

Ideally, then, the algorithm then finds the best assignment of clusters and centroids to explain the variance in the data. However, since the algorithm is non deterministic and depends on the initialization of the data, it can arrive at local minima and should be run several times to ensure good performance.

5 Experiments/Results/Discussion [1 - 3 pages]

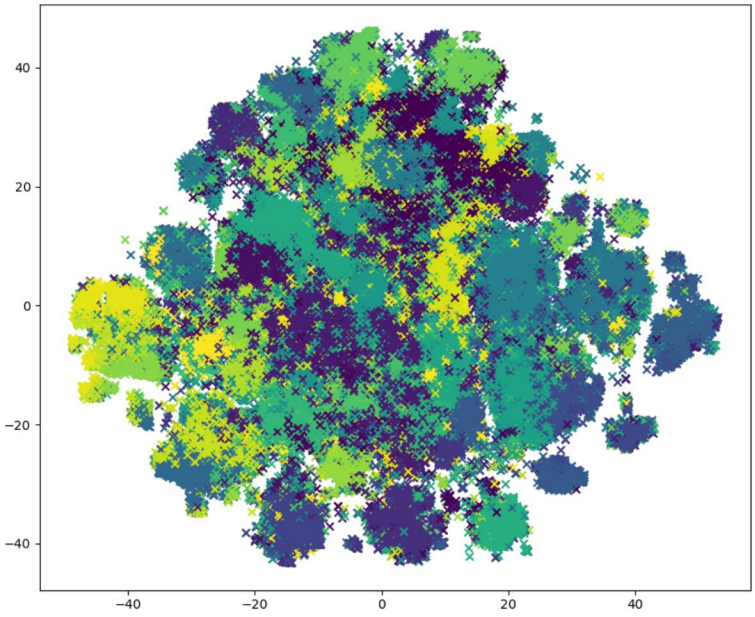
To tune the models, I needed to choose the number of clusters to choose for the k-means algorithm. A common method of choosing this is elbow testing, which

is both a quantitative and qualitative method. This involves running multiple trials on the data with different numbers of clusters and evaluating the model at the conclusion of each trial while incrementing the number of clusters afterward.



The results of this are as follows:

Accordingly, I chose to use 50 clusters, as we start to notice diminishing returns at this point. Qualitatively, results from the **k-nearest-neighbors** algorithm were quite encouraging. When given a Cabernet Sauvignon from California, the model returned four other California Cabernet Sauvignons and one Cabernet Sauvignon from Washington. Even further, when given an Oregon Pinot Noir, the model returned five other Oregon Pinot Noirs as its closest neighbors. However, the model seemed less accurate when I fed less detailed descriptions of wines. When given a short or incomplete review, the model produced recommendations with significantly more variance, including recommending red wines when given a white or sparkling wine. This indicates that adding some quantitative data could boost the performance of the model. In two dimensions, we can see the $k = 50$ clusters produced:



6 Conclusion/Future Work [1 - 2 paragraphs]

In conclusion, this approach is encouraging. The main drawback I noticed was the difficulty in interpreting and evaluating the results produced by the model. For example a labeled dataset would be much easier to evaluate as I could more directly measure the accuracy of the groups. But I think the approach is more engaging and intuitive than other methods. For example, when I input several attributes of a hypothetical wine that I prefer such as `blackberry, black tea, cedar, tannic` the program was able to recommend several wines in the data which seemed reasonable.

7 References/Bibliography (No page limit)

From Sci-Kit-Learn: TSNE, KMeans, K Means, K Nearest Neighbors, TF-IDF
From Natural Language Toolkit: Stopwords, Porter Stemmer