Wine Recommender

A web application to search for new wines based on known preferences

Ezana Tesfaye
CMPE 272, San Jose State University
ezana.tesfaye@sjsu.edu

Lishan Zhu
CMPE 272, San Jose State University
lishan.zhu@sjsu.edu

Shengqi Suizhu
CMPE 272, San Jose State University
shengqi.suizhu@sjsu.edu

Abstract--Making a choice in the wine aisle will never be a challenge again. A user will be able to input a name or keyword from a wine they like, and set filters for maximum price and wine pairing. The application outputs recommendations based on preferences.

Keywords--recommendation, sentiment, similarity, wine, preference, search, filter

I. INTRODUCTION/BACKGROUND

Sometimes when people find a type of wine they enjoy, they are likely to try similar types of wine. However, wine isles can often be overwhelming with the number of choices there are. This project aims to help a user narrow down wine choices. The application then recommends new wines to the user that it thinks they would like.

This document explains what goes on from the point a user inputs a wine, and how the web application searches and finds results. The user first types in a wine varietal or specific wine name, sets a maximum price that filters the results, and then chooses one or more foods to pair the wine with. A wine is a valid pairing as long as one of the user's specified foods matches with at least one of the wine's specified foods. The top nine wine matches are returned in a 3-by-3 grid once the user submits the search parameters. Each match shows a photo of the wine, the wine's name, price, region, and foods it pairs with. Finally, the user can save the wine to their account if they are interested in trying it in the future.

II. USER GUIDE

The user will be prompted with the homepage. The user can search any wine without signing in or even without an account.

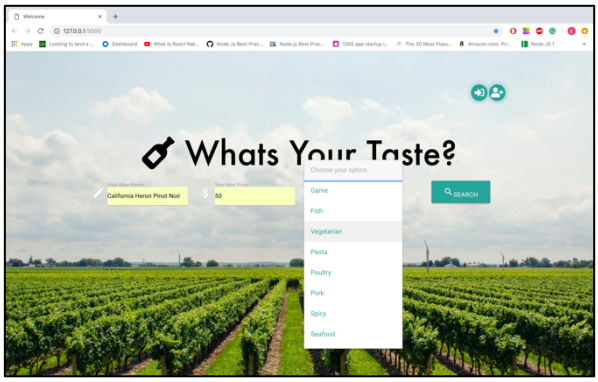

Figure 1. Landing Page

Name, vineyard, varietal, price, pairings, and a photo are shown to the user in the results.

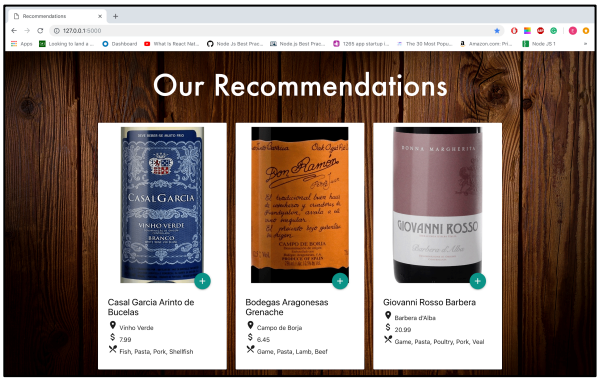

Figure 2. Search Recommendations

Users can save their search if they have an account. When they are not logged in, pressing the add button will take them to the log-in page.

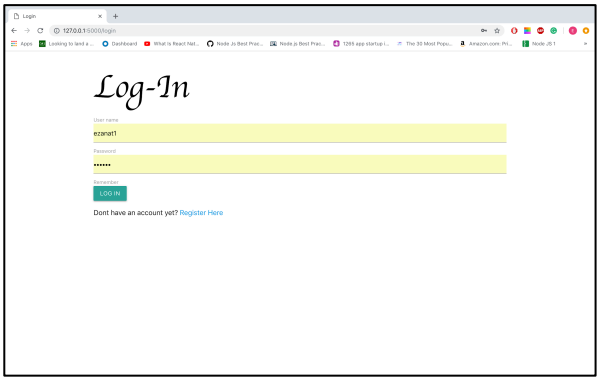

Figure 3. Log-in Page

The user can log in if they have an account, or they can click "Register Here" to register for an account.

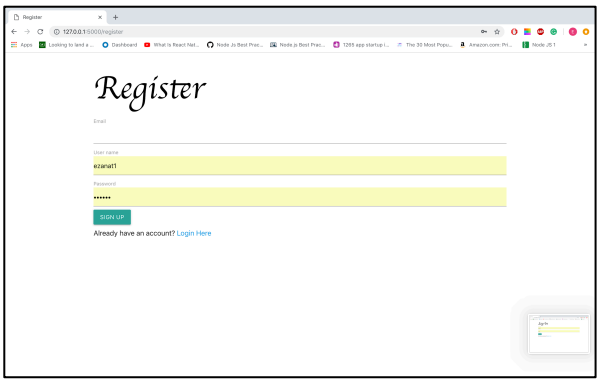

Figure 4. Registration Page

Once logged in, the user can search for wines in the same way they were able to do so on the landing page. They now can save the wine.

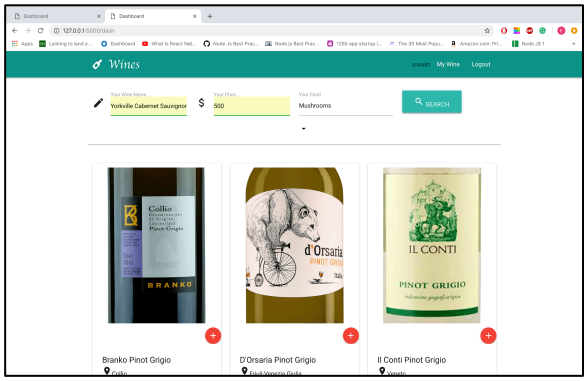

Figure 5. User's Homepage

The user can access saved wines by clicking on the "My Wine" tab. There is also an option to delete the wine from that page.

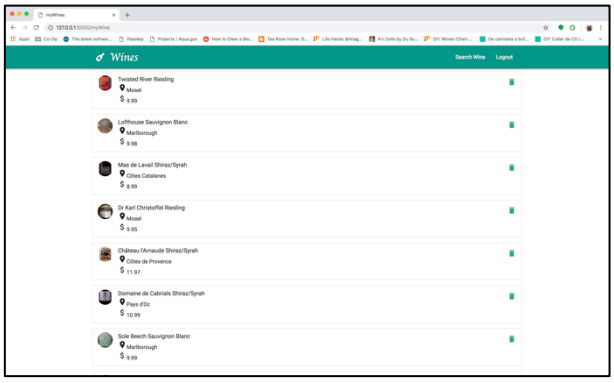

Figure 6. User's Wine List

III. DESIGN CONSIDERATIONS

The wine recommendation system is designed for people that are not wine experts, but still do understand the difference between wine varietals and regions. They should at least know one wine name, varietal, or vineyard that they know they like. Based on their preference, they will get more wines

The maximum price filter allows the application to recommend wines to people of all budgets. A typical college student may have different expectations for how much a bottle of wine should cost than a middle-aged person hosting a party for friends. Additionally, it works well with people that are working on a strict budget because the user can easily calculate a per bottle budget and enter it into the tool.

The wine pairing was added to satisfy the situation the wine would be used in. Some foods are better paired with certain wines. For example, a general rule of thumb is red meat pairs with red wines, and white meat pairs with white wines (though there are definitely exceptions).

Saving wines was built with the idea that most users will not buy many wines at once. The user may want to save bottles they were not able to purchase for another trip. That is why user registration and login was implemented, and users can save wines from previous searches to their profile for later access.

IV. IMPLEMENTATION

A. Data Collection

Data for the recommendation system was extracted using a web crawler from the Vivino website. All of the wine information can be accessed through an API that is formatted uniformly for all wines. The crawler iterates through a list of wine id's and parses each website request's output json. From the json, the following attributes were extracted: average rating, price, varietal, vineyard, region, and the first 50 pages of comments for wines that had more than that.

B. Recommendation Algorithm

The basis for the wine comparison algorithm is vector comparison. We treat each wine in the collected data as a vector which features multiple dimensions such as price, rating, region etc. In order to measure the difference between two vectors, we are essentially measuring the distance between them. After some trial and error, an L2 norm, euclidean distance is best measured to calculate the distance. And to find the wine that is closest to the user input, we simply need to calculate the distance from the target wine to the rest of the wines and sort out the closest. With the Numpy linear algebra package, such calculation is easily completed.

The distance measurement can be customized and further extended to filter different results. Most numerical data such as price, rating are normalized to a 0 to 1 range, different normalization methods can be used such as a simple rescale or a sigmoid function to remove extreme results.

For categorical data such as region, wine variance, it would not make sense to simply assign some arbitrary numerical values to them, thus we compare the two wines categorical data, if they are identical, a 0 is assigned, else the distance is 1 to indicate that they are different types of wine.

Different filters can then also be applied to the comparison method by adding or subtracting features needed to compare in the distance calculation method. With this, we are able to generate pairing wines for different food, price, etc.

The sentiment value for different wine is also analyzed in order to discover an additional vector, with the help of

TextBlob package, we can measure the different sentiment for different comments for each wine. After aggregating all the comments, we can generate an overall sentiment rating for the specific type of wine.

In the database, all wine is stored with their detailed information, and with the user database, we are also able to let the users store their favorite wines and later display it on other pages.

C. Web Framework

We used flask for our back end which process the requests sent by ajax from the html pages. So when the user types in a wine it will be received by our routes in flask. The routes call our machine learning algorithm class for similar wines and give us list of option. Then we again send back the data we receive from our machine algorithm to our displays.

V. CONCLUSION

The main motivation behind our product was to help someone hosting a party with making a wine decision. Features in the web application address the main concerns of a user before they choose a wine. After understanding what a user's thought process before choosing a wine is, we implemented filters that would give the user a better recommendation. This saves users time when they are picking out wine, and gives them an educated recommendation instead of having them randomly choose a wine.

VI. REFERENCES

[1] Vivino
https://www.vivino.com/

[2] 15 Rules for Great Wine and Food Pairings
https://www.foodandwine.com/slideshows/15-rules-great-wine-and-food-pairings

[3] Creating Forms
https://flask-wtf.readthedocs.io/en/stable/form.html

[4] Flask, Web Development, One Drop at a Time
http://flask.pocoo.org/