Sommelier:
A personal wine recommendation engine[1]

1: Project step III updated.

ANLY-605 (Fall 2022) at Georgetown University

Jingsong Gao*
Ercong Luo [†]
Rui Qiu [‡]

October 7, 2022

_____

* jg2109@georgetown.edu
[†] el890@georgetown.edu
[‡] rq47@georgetown.edu

Contents

List of Figures

List of Tables

1 Project Description

1.1 General Description

With our app Sommelier, an end user can find recommendations of wine region and grape varietals based on a few keywords of flavor notes. This saves them the time, money and liver to try many bottles of wine that they don't like.

1.2 Project Deliveries

An interactive web application where the end user inputs a description of their wine preference, and the application surfaces a list of top 5 recommendations of `<wine region, grape variety>` pairs. The back-end of the application is where natural language processing and machine learning algorithms are deployed to surface recommendations at the highest accuracy possible.

1.3 Out of Scope

This project will not include any background educational knowledge on wine tasting. Nor will the application have any mechanisms for collecting user feedback in the first release.

1.4 Assumptions

We assume that:

1. The expert wine reviews in the training dataset are unbiased and accurate descriptions of wine flavors.
2. The primary factors that influence the flavor of a wine are the region of production and the grape varietal, hence the choice of surfacing `<wine region, grape variety>` pairs as recommendations to an end user.

1.5 Constraints

For the constraints, we have:

▶ **Biased Review:** The reviews we will use to train the classifier, especially, the description of notes, are subjective. The reviewers are also anonymous. Amateur and professional tasters might use different words to describe the same wine.
▶ **Outdated Wine Data:** The wine dataset is updated on November 22nd, 2017, so any wines produced after the date will not be recommended.
▶ **Incompatible Target:** The classification model is trained to provide a tuple `<wine region, grape variety>` that matches the description of a review. But since recommendation to users is an open-ended question, it will be difficult to evaluate the "goodness" of recommendations until there is enough collected user feedback for product analysis and review.
▶ **Limited Computing Resource:** Cloud service providers have limited computing resources for free deployment, especially the RAM:

  • For Heroku, the free Dyno has 512MB RAM.

- For AWS EC2, the low-end t4g.nano instance (0.0042 USD/h) also has 512MB RAM.

Considering the server handler to process requests and other resources need to be loaded into the memory, models have to be smaller than 200 megabytes altogether. This could prevent us from using most deep learning-based solutions.

▶ **Discrepancy Between Descriptions and Reviews:** We will use text-based reviews as proxies for a user's description of their wine preference. Admittedly, this will now account for the variance in user input in terms of word choice preference, sentence lengths, and subjective bias that cannot be captured by our language model.

1.6 Development Roadmap

Because most NLP models today are based on transformers, and our early thought for this project is that a transformer-based model is going to outperform a traditional model like SVM or tree-based ensemble learning. That being said, we still followed class requirements and trained models such as an SVM, a perceptron, and a naive Bayes classifier from sklearn and constructed a stacked model. We included evaluations for both the transformer model (BERT-mini) and the stacked ensemble model.

Items in bold are listed as course-required stages, while the ones in italic are extra milestones.

▶ **Stage 1: Problem Identification and project charter. Submitted by Sept 7, 2022.**
- Create a repository for the project and set up the developing environment. Done by Sept 14, 2022.
- Initial training of the candidate model "Wine BERT". Done by Sept 21, 2022.

▶ **Stage 2: Solution by generative method (this report). Submitted by Sept 28, 2022.**
- Model performance comparison: Wine BERT vs generative methods (included in the report). Submitted by Sept 28, 2022.

▶ **Stage 3: Initial deployment. Submit by Oct 12, 2022.**
▶ **Stage 4: Solution by hyperparameter optimization. Submit by Oct 26, 2022.**
▶ **Stage 5: Complete project. Submit by Nov 16, 2022.**

2 Introduction

Whenever we talk to a wine connoisseur, we will hear them talk about their wine preferences based on two primary factors: the region where the wine is from and the grape varietal from which the wine is made. It is obviously a steep learning curve for a newcomer to know their personal wine preferences in terms of these two primary factors because it requires trial and error that most of us don't have the money, time or the liver to do. But what if we have our personal digital wine steward, or a sommelier as the French call it? That is what our app is for.

Sommelier App takes advantage of natural language processing and machine learning algorithms, trained using a dataset of 117283 wine reviews written by expert wine tasters to learn the differences in described flavor notes that differentiate a dataset of over 500 `<region, grape varietal>` pairs.

3 Analysis of the Dataset of Trained Model

3.1 Exploratory Analysis

Since our dataset is text-based, the exploratory data analysis (EDA) procedure is rather limited. A typical sample of data is listed in Table 1.

	Description	region_variety
0	The deep red-black color reaches...	France-Languedoc-Roussillon:Cabernet Sauvignon
1	Complexity and varietal character...	US-California:Merlot
2	Rhubarb and cranberry fruit, along...	US-Oregon:Pinot Noir

Table 1: Two samples in raw data.

After preprocessing, the sample data we use to train the model are listed in Table 2.

	keywords	region_variety
0	core adequate acidity moderate...	265
1	complexity varietal character black plum...	17
2	rhubarb cranberry fruit red apple light simple...	4
3	impressive fullness ripeness black cherry leat...	251
4	dusty tones mineral saffron pollen concentrate...	11

Table 2: Five pre-processed samples. *(Descriptive keywords are extracted from raw reviews, region_variety pairs are converted to numeric labels)*

There are in total 584 region varieties in the dataset. A slim version of 11680 records are included as the training set of all candidate models below.

The keywords are vectorized into a feature vector of length 6081, and a slice of descriptive features used in wine reviews are:

```
Vectorizer features: ['bell', 'bellangelo', 'belzbrunnen', 'benito',
'benjamin', 'berenguer', 'beresan', 'bergamot', 'bergerac', 'bernard',
'berried', 'berries', 'berry', 'berryish', 'berrylicious', 'bertani',
'best', 'better', 'betz', 'beverage', 'bianca', 'bianco', 'biancolella',
'bical', 'bieler', 'bienenberg', 'big', 'bigger', 'biggest', 'bigtime',
'bilberry', 'bill', 'billards', 'billing', 'billo', 'bing', 'biodynamic',
'biodynamically', 'birch', 'bird', 'birds', 'biscotti', 'biscuit',
'biscuits', 'biscuity', 'bisquertt', 'bistro', 'bite', 'bites',
'biting', 'bitner', 'bits', 'bitter', 'bitterness', 'bitters',
'bittersweet', 'black', 'blackberries', 'blackberrry', 'blackberry',
'blackcurrant', 'blackened', 'blackness', 'blacktop', 'blanc',
'blanca', 'blanched', 'blanco', 'blancs', 'bland', 'blangé', 'blasting',
'blatant', 'blaufränkisch', 'blaye', 'blazing', 'blend', 'blended',
'blending', 'blends', 'bleue', 'blind', 'bliss', 'blockbuster',
'blockier', 'blocks', 'blocky', 'blood', 'bloody', 'bloom', 'blossom',
'blossoms', 'blossomy', 'blowsy', 'blue', 'blueberries', 'blueberry',
'bluebery', 'blush', 'board']
```

3.2 Baseline Model: BERT-mini

Our baseline model is a little bit "unorthodox". We use the BERT-mini model with the raw review texts as inputs. There are several reasons for this:

► **Pre-knowledge**
  BERT models are pre-trained on a large corpus of multi-domain text and learned a lot of general language knowledge. We only need to fine-tune the model by training it with a much smaller dataset specific to our task.

► **Powerful Text Tokenizer**
  BERT models have a strong text tokenizer with a vocabulary of 30k words and a word-piece strategy to handle out-of-vocabulary words. This means we don't need to do any preprocessing (such as punctuation cleaning, keywords extraction, etc.) on our text data.

► **Low Computing Cost**
  BERT series are very large models and require a lot of computing resources (CPU, memory, GPU). We use a mini version of BERT that is only 50MB in size and is able to run relatively fast with limited computing resources.

On a single GPU, it took 35 minutes to finetune BERT-mini on the full training data and 7 seconds to evaluate the model on the full test data. The top-5 accuracy is 92.0% on the training data and 83.4% on the test data.

3.3 Metric: Top-5 Accuracy

In classification tasks, we usually use **accuracy** (or top-1 accuracy) as a metric for model evaluation. An evaluation is considered accurate only when the model prediction and the true label are identical.

By **top-5 accuracy**, we meet that an evaluation is considered accurate if the correct label is within the top-5 predictions by posterior probability. So a 92.0% accuracy in our case means that 92.0% of the time, the correct label is within the top-5 predictions on a given input. This top-5 accuracy metric is chosen because:

► **Similarity between wines**
  When a user types in a description of a wine, the model should be able to return a list of top-5 recommended wines. The user can then choose the one that best matches his/her taste.

► **Generalization of the model**
  By using a relatively lenient top-5 accuracy, we gave the model more space and tolerance to learn the similarities between wines. The model was taught to understand a one-to-many relationship from reviews to wines, rather than the strict one-to-one relationship when using the top-1 accuracy.

► **Alignment with usage scenarios**
  The usage scenario of our application is to recommend the top-5 relevant wines to users according to their descriptions. Using top-5 accuracy helps the model to be consistent during training and inference.

4 Model Selection

4.1 Individual Models and Stacked Model

Since we are selecting candidate models for stacking, and ultimately competing with BERT-mini, we have decided to start from basic classification models ranging from Multinomial Naïve Bayes, Perceptron, Linear Support Vector Machine Classifier, Decision Tree Classifier, and Logistic Regression Classifier as our candidates.

Figure 1 demonstrates the classification accuracy (top-5 accuracy) on the testing set, and we can see that Logistic Regression, Naive Bayes, and Linear SVM are 3 top candidates. Don't be fooled by the accuracy barely past 0.5, it's already pretty impressive considering the problem we are tackling is to find the top 5 recommendations from over 500 wines that fit a verbal description. And this is achieved even without hyperparameter tuning.

The strategy and sequence we used to find the right candidates can be summarized in two sentences:

1. Level 0 models should be ordered from best to worst in terms of performance. An analog is to use stepwise linear model selection from a null model. Specifically, we add the model with the best performance, then the second best to see if it improves, etc.
2. Level 1 models (meta learner) don't need to be complex models. Given that level 0 models are already powerful models, the meta learner is simply utilized for aggregating and calibrating predictions from the base learner.

We also would like to recycle the Logistic Regression Classifier as a level 1 model (meta learner) to aggregate the results of the level 0 models.

We trained two combinations of stacked model following the naming convention below:

- ▶ We use the initials of the level 0 candidates, and stitch them together in a sequential order. Then we connect the letters with the initial of the level 1 model using an underscore.
- ▶ In this manner, we have `stacked_ln_l` stands for a stacking classifier with a logistic regression classifier + a naïve Bayes classifier as level 0, and a logistic classifier as level 1.
- ▶ And `stacked_ls_l`, a stacking classifier with a logistic regression + a linear SVM as level 0, and a logistic classifier (again) as level 1.

We ruled out the ensemble models in stacking, since the training time of ensemble models in our case is comparatively long due to the size and feature number of the dataset.

4.2 Model Performance Evaluation

From the previous boxplot, we learned that the stacking method did not take the model performance to the next level. The performance of BERT-mini is still far better.

If we compare the confusion matrices of `stacked_ln_l` versus BERT-mini side-by-side (Figure 2), the BERT-mini has brighter-colored pixels concentrating on the diagonal line.

Note that both diagonal lines seem to be in greenish-blue color, however, the actual values are close to 0.5 (stacked model) and 1 (BERT-mini)
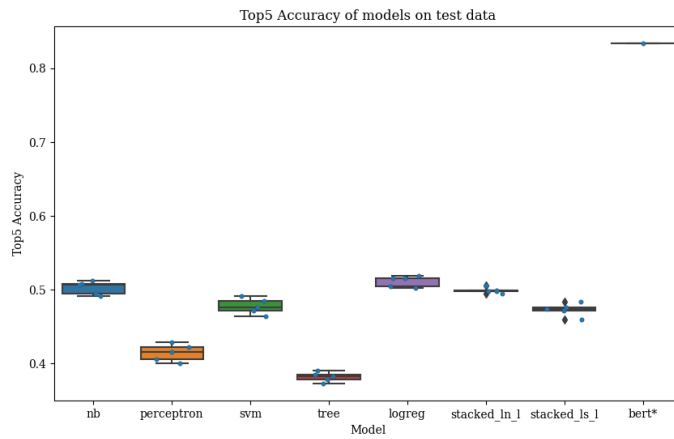
Figure 1: Top5 Accuracy for models on test data. *Note: BERT model only has one data point as cross-validation was not performed.*
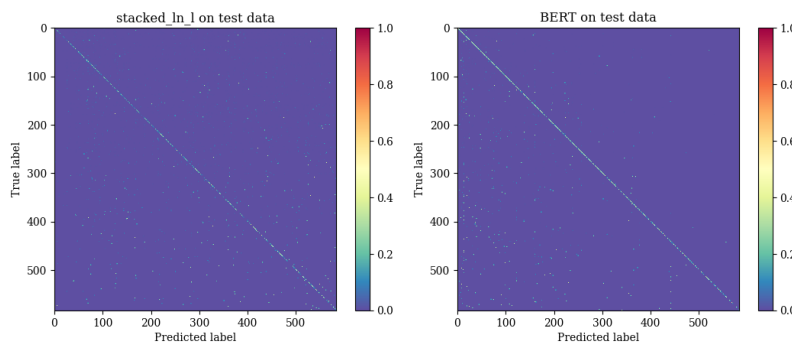

Figure 2: Confusion matrix of the stacked model `stacked_ln_l` and BERT-mini on test data.

respectively. This is caused by the interpolation method for `imshow`. Basically, the color values of each pixel are averaged to its neighbors because the original heatmap is too big to display on a small figure.

Another takeaway from the confusion matrices is that BERT-mini has a tendency to predict the response to be within class 0 and class 200. However, this is more like a coincidence as the numeric value in class is nothing but an index, which is not ordinal.

Bear in mind that we haven't done any hyperparameter tuning in either stacked model or BERT, plus the performances in training dataset imply the existence of potential overfitting.

4.3 Cost Estimation

The last two aspects we would like to investigate are the fit time (Figure 3) and inferencing time (Figure 4), which are used to imitate the runtime of training and actual prediction.

Surprisingly, BERT-mini does not top the fit time in our test run. Of course, the five basic candidates are fast, and we have to admit that the logistic regression classifier is not bad in performance as well. Even though we managed to control the complexity of those two stacked models, they generally took more time to train and gave out unreliable predictions.

In practice, when we set a periodic batch job scheduled, say it's designed to train the model with new data, the more time we need to train the model, the higher computation resources we need to fulfill such a task. And this can be translated into higher cost for the company.

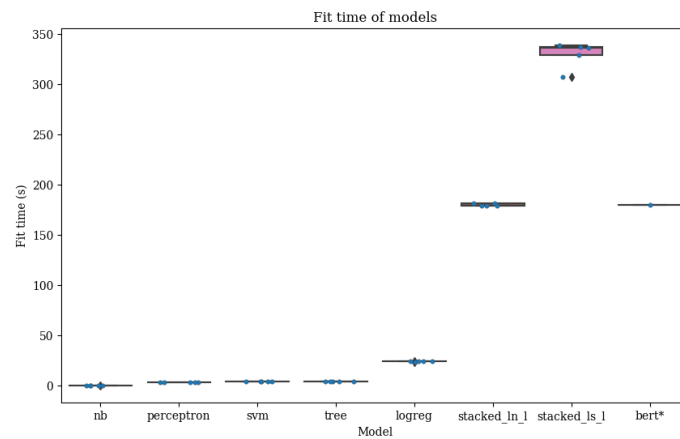We run the inference sequentially over 1000 samples and calculate

Figure 3: Fit time(s) for models on training data. *Note: the fit time for BERT model is estimated proportional as it was trained on the full training data.*
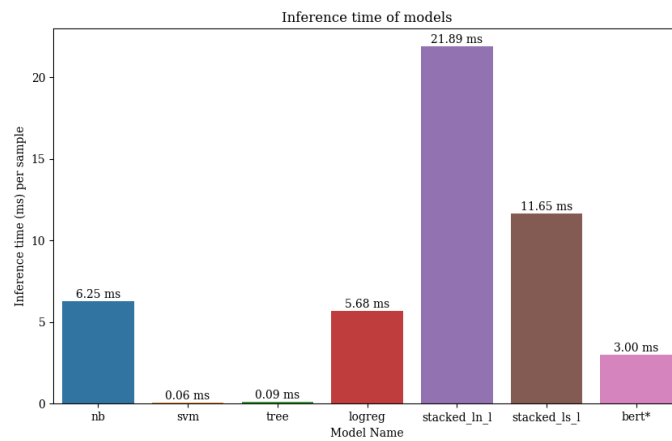

Figure 4: Average inference time(ms) per sample for models.

the average inference time per sample. Similarly, BERT-mini is not the fastest one either. The inference time in production can be translated into waiting time for each end user, and 3 milliseconds per request is blazingly fast. Nevertheless, we should be aware of the risk of large amounts of concurrent requests in an imaginary scenario. Making the application a real-time experience is a key optimization goal for our project.

5 Initial Deployment

5.1 Overview

The initial deployment of Sommelier is a web application that allows users to input a wine description and get a wine recommendation. The web application is built with Flask, React and Semi Design.

The logo is designed in Figma[1], and the avatar is generated by DALL-E 2 with a prompt of *"A sommelier, van Gogh style"*.[2]

In terms of responsiveness, we did not make many changes to the existing Create React App template. However, we did make some adjustments to the layout of the web application to make it more mobile-friendly. The body of the text will adjust its width based on the viewport size of the device. A brief mockup of the mobile version is shown on the right hand side, or you could simply view the app on your device.
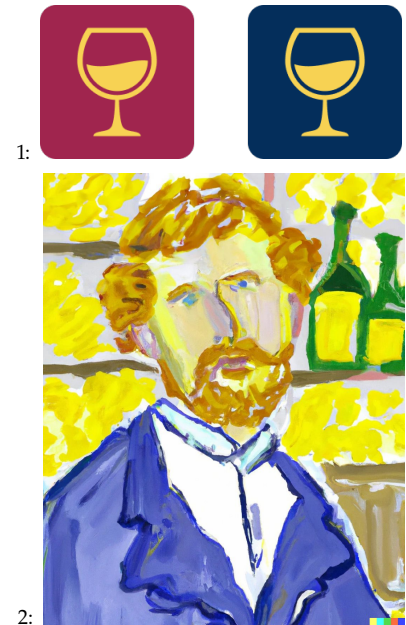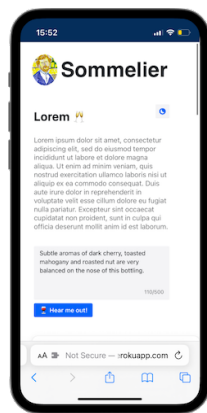

1:

2:

Figure 5: A mock up on iPhone 13.

In addition, we believe it is vital to support dark mode for web applications. A user could easily switch between light and dark modes with a click on the top right corner.
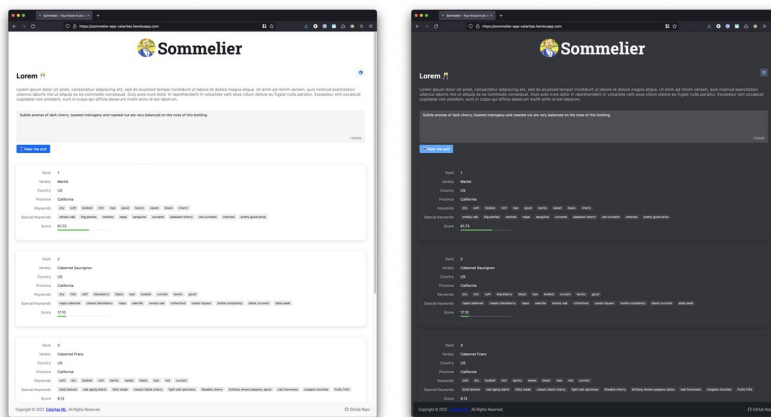

Figure 6: Snapshot of app webpage. *Dark mode supported.*

5.2 Input box and Recommendation Cards

The core of this website can be summarized as two components: **the input box** and **the recommendation cards**.

The input box is a text area that allows users to input a wine description. It is self-explanatory. Some of the example input that a user could try are shown below:[3]

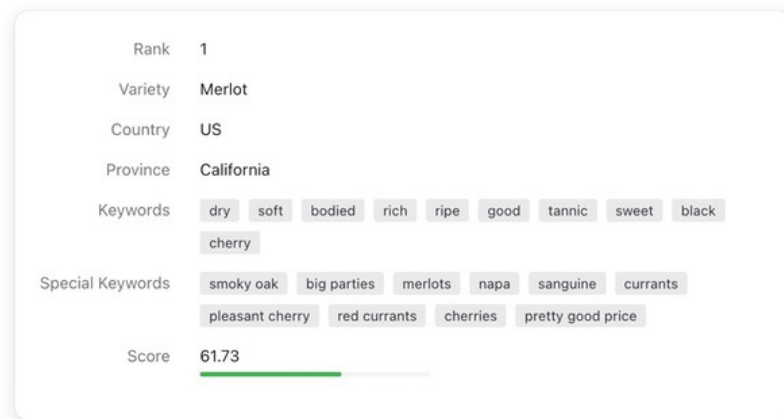3: And of course, users are more than welcome to try their own inputs.

Figure 7: The header and an input box.
Note: the prologue is placehold by lipsum.

1. Pronounced aromas of salinity, petrol and toasted oak rise up out of the glass. This wine offers a rich, dense texture accented by fresh apple notes and a slightly petrol finish. This is a wine of intensity and body, although it does carry a bit of alcohol heat on the finish. Pair it with mild pasta dishes, fish or cheese, though, and you won't even notice.
2. Sometimes you stick your nose in a glass of wine and it just smells like Riesling, pure and true. This wine carries vibrant aromas of white blossoms, grapefruit flesh and wet stones. On first sip, it's a harmonious blend of grapefruit, lemon and a bit of apricot that seems to add body in addition to complexity. A wave of acidity wipes the palate clean, allowing mineral and petrol notes to shine through. This is a wine that will only get better over the next few years.
3. Petrol and eucalyptus dominate the nose on this, but juicy fruit aromas are present and inviting. Weighty and with a silky ripple, this is an intensely flavored, savory wine. Preserved lemons, bay leaves, wet rocks and petrol hang together in a sophisticated package that ends with firecracker acidity. Both serious and refreshing, this wine is ready to take on just about any meal-pairing situation."
4. Heady aromas of clove, cinnamon and peppercorn ride high above the brandy-soaked black cherry core in this hedonistic red. It's full in feel on the palate, with accents of cola, clove and cinnamon framing the plump dark-cherry core. Polished tannins and integrated acidity keep it grounded and bright, making this a great red for those who love rich fruit and oak in tandem.
5. This spicy red displays aromas of Sichuan peppercorn, blood orange peel and mentholated tobacco on the nose, with a solid core of black cherry and red plum. It's polished and supple on the palate, with smooth tannins and pulsing acidity working in tandem to lend a firm webbing for the plump cherry and plum flavors to shine. Accents of orange peel and purple flowers lend levity, with a savory twang of granite on the close.

The other component are the model-generated top 5 predictions. The recommendation cards are designed to be intuitive. The cards are arranged in a grid layout, with related metadata about that wine. A colored progress bar also indicates how confident the model is about the

prediction.[4]

Rank	1
Variety	Merlot
Country	US
Province	California
Keywords	dry soft bodied rich ripe good tannic sweet black cherry
Special Keywords	smoky oak big parties merlots napa sanguine currants pleasant cherry red currants cherries pretty good price
Score	61.73

Figure 8: A recommendation card.

4: Note: The assignment requires us to include: (1)"Graph of the dependent variable with any variable of importance"; (2)"Graph(s) from above with an update of prediction (image generated when we predict a feature vector)".
We believe in our case, an example of the prediction made in a recommendation card would suffice as both of the requirements above since it includes
(1) the dependent variable (region-variety) and important features (keywords, special keywords) that the model actually looks at,
and (2) the prediction made by the model.

5.3 Heroku Application

The Platform, and the Repository

The web application is deployed on Heroku*. The repository is hosted on GitHub† under the organization CeleritasML.

Heroku is the most convenient platform to host a web application with free tier available and backend support. Users can go and test the model. The example inputs (descriptions) are provided in previous subsections, but not required.

Deployment, debugging, and updates

Auto CI/CD is implemented in this Heroku application. Whenever a new commit is pushed to the main branch, Heroku will automatically build and deploy the application. The only downside is that the free tier only supports one single user to be the owner of this application. Thus, whether commits by other team members would be deployed is in doubt. Still, it is not a big issue for now.

Starting from now, all of our updates will be synchronized to the web application through continuous deployment.[5]

5: Sadly, Heroku is cancelling its free tier from November 28, 2022.

* Application URL: https://sommelier-app-celaritas.herokuapp.com/
† Repository URL: https://github.com/CeleritasML/sommelier-app