BottleWise: An Interactive ML System for Alcohol Recognition with Human-in-the-Loop Correction

Interactive Machine Learning Course Report

1 Introduction

Identifying an unfamiliar alcoholic beverage is a concrete challenge for travelers, international students, and immigrants who encounter products whose labels are written in a language they cannot read. Standard machine-learning pipelines address this with a single-shot prediction. This output often arrives without explanation and gives the user little way to respond if it is wrong. We argue that this is especially unsuitable in the alcohol domain: the consequences of a misclassification are asymmetric. Mistaking a non-alcoholic drink for an alcoholic one (false positive) is an inconvenience; mistaking an alcoholic drink for a non-alcoholic one (false negative) carries genuine safety risk, particularly for underage users.

BottleWise departs from the black-box paradigm by placing the user at every decision point. Rather than returning a single answer, the system walks the user through a multi-step process: the model predicts, exposes its uncertainty, and waits for the user to inspect and accept or reject the result before any downstream action, such as querying a large language model (LLM), is taken. Incorrect predictions are corrected in-app, and the corrected examples are immediately used to retrain the classifier, closing a human-in-the-loop feedback cycle within the same session.

The application runs entirely in the browser using Marcelle [3] and TensorFlow.js, requiring no server and no data upload.

2 Scenario and Users

2.1 Context of Use

The primary context is *in-store or on-the-go*: a user photographs a bottle or can, and the system classifies it while communicating uncertainty in plain language. The workflow is intentionally short (six wizard steps) to fit a transient, mobile-style interaction in a shop aisle or at a restaurant table.

2.2 Primary Users

The primary users are **non-native speakers**, including international students, tourists, and immigrants, who can photograph a label but cannot read it. Their domain expertise in alcohol is low to moderate; their motivation is practical ("is this safe to drink?", "will I like this?"). A critical sub-group is **underage users**: because they may be unaware of local alcohol regulations, a false negative (a mislabeled alcoholic drink) is a high-severity failure that must be flagged explicitly regardless of model confidence.

2.3 Secondary Stakeholders

Retailers benefit from customers who can independently navigate product categories without staff assistance. HCI researchers and educators may use the system as a case study of transparent, uncertainty-aware interactive ML.

2.4 Relevance to Interactive ML

Prior work on interactive machine learning [2] emphasizes that non-expert users can improve model accuracy when given simple, structured feedback mechanisms. Settles [5] showed that active learning, which selects informative examples for labeling, substantially reduces the annotation burden. Amershi et al. [1] argued that involving users throughout the interactive machine-learning loop improves both system effectiveness and user experience. Our design follows these principles: the KNN correction loop is a lightweight active-learning mechanism that adds the most informative example at each session (the one the model got wrong).

3 Application Design

3.1 Workflow

The application is organized as a six-page *wizard* (Fig. 1). Each page corresponds to a discrete step; the user must complete each step before advancing.

P1. Taste Profile (Welcome). Three binary preference selectors (Strength, Sweetness, Flavor) are auto-saved to `localStorage`. Values are injected into the Gemini prompt at enrichment time.

P2. Label Examples (Training). The user uploads images and assigns labels from a 26-class vocabulary (whiskies, wines, beers, spirits, Asian spirits, and a dedicated *Non-alcoholic* class). Each image is passed through MobileNet v2 *as a feature extractor*: the network's final classification head is removed, and the activations of the penultimate layer are read out instead. The result is a **1280-dimensional numeric vector** — a compact, fixed-length description of the visual content of the image (shape, texture, colour gradients, edge patterns) that MobileNet learned to produce during pre-training on 1.2 million ImageNet images. Visually similar images produce numerically similar vectors; this is the property the KNN classifier exploits. These vectors, together with the user-assigned labels, are stored in a `dataset` backed by `localStorage`.

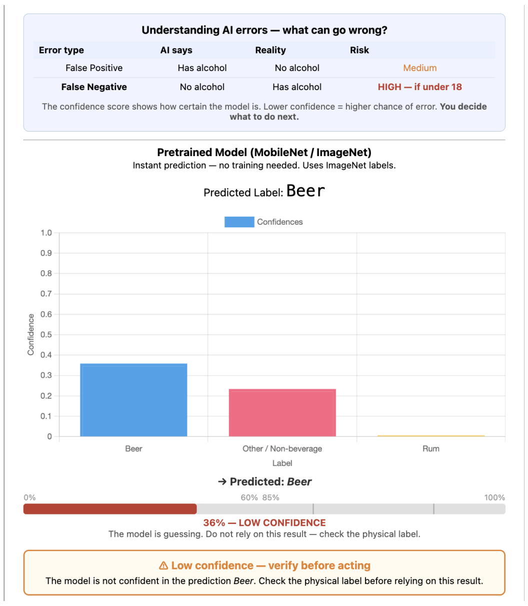
P3. Train the KNN. A $k = 3$ nearest-neighbour classifier is trained on the labeled 1280-dimensional vectors. At inference time it finds the three stored vectors closest to the query image's vector (by Euclidean distance) and majority-votes their labels. A `trainingPlot` widget gives live feedback during training. This step is optional:

the pretrained MobileNet path requires no prior training.

P4. Identify a Beverage. The user uploads a photo and chooses between two classifiers: *Quick Identify* (pretrained MobileNet, ImageNet labels) or *Custom KNN* (user-trained labels). Both produce a prediction immediately.

P5. Review Results (Uncertainty Visualization). Both models' outputs appear on a single page with confidence meters, tier messages, and a false-negative / false-positive explainer table visible at all times. The LLM is *not* called at this stage.

P6. Decision and Enrichment. Three action buttons (*Accept*, *Flag as Uncertain*, *Retake Photo*) give the user full agency. Accepting triggers the Gemini enrichment (only when confidence $\geq 60\%$ and a non-"Non-alcoholic" prediction). An always-visible correction panel lets the user supply the true label; the KNN retrains immediately upon submission.



Understanding AI errors — what can go wrong?

Error type	AI says	Reality	Risk
False Positive	Has alcohol	No alcohol	Medium
False Negative	No alcohol	Has alcohol	**HIGH — if under 18**

The confidence score shows how certain the model is. Lower confidence = higher chance of error. **You decide what to do next.**

Pretrained Model (MobileNet / ImageNet)
Instant prediction — no training needed. Uses ImageNet labels.

Predicted Label: Beer

→ Predicted: *Beer*

36% — LOW CONFIDENCE
The model is guessing. Do not rely on this result — check the physical label.

⚠ Low confidence — verify before acting
The model is not confident in the prediction *Beer*. Check the physical label before relying on this result.

Figure 1: Step 4 *Review Results*: confidence meter (color-coded low/medium/high zones), false-negative alert panel, and the FP/FN explainer table. Both model results are shown side-by-side before the user makes a decision.

3.2 Uncertainty Visualization

The Review page (Step 4) is the core contribution of the system. It surfaces three layers of uncertainty information that non-expert users can act on:

- **Confidence meter.** A three-zone progress bar (red / orange / green) spanning $[0\%, 100\%]$ with vertical dividers at the 60% and 85% thresholds. Color and a plain-language label (LOW / MODERATE / HIGH confidence) communicate the zone without requiring the user to interpret raw probabilities.
- **FP/FN explainer.** A static table present on every visit to Step 4 defines *false positive* and *false negative* in domain language ("AI says Has alcohol / Reality: No alcohol") with a severity column that calls out the higher risk of false negatives for under-18 users.
- **Alert panels.** Dynamic panels, rendered in red for false negatives and orange for low-confidence non-false-negative predictions, appear directly below the confidence meter. The false-negative panel includes step-by-step instructions for manual label verification (look for the ABV percentage on the bottle).

3.3 Active Learning Correction Loop

The correction panel (always visible on Step 5) implements a minimal active learning loop:

1. User selects the correct label from the full 26-class vocabulary.
2. The current prediction image is re-processed through MobileNet v2 to extract its feature vector.
3. A new labeled example is appended to `trainingSet` in `localStorage`.
4. The KNN classifier retrains synchronously in the browser.

The latency for steps 2–4 on a modern laptop is under one second, making the feedback loop feel immediate.

3.4 LLM Enrichment

Accepted predictions invoke the Gemini API with a structured prompt that requests: category, origin, typical ABV, description, food pairings, and a safe-consumption note. The prompt includes the taste profile as a natural-language preamble ("The user prefers mild, sweet, and fruity drinks"), causing the description and pairings to reflect the user's stated preferences. A model cascade (`gemini-2.0-flash` → `gemini-2.0-flash-lite` → `...`) handles rate-limit (HTTP 429/503) responses transparently.

4 Implementation

4.1 Stack

The application is a single JavaScript file (`marcelle-app/src/index.js`, ≈900 lines) built on the Marcelle framework [3] v0.6.5. All ML inference runs in the browser via TensorFlow.js; no data leaves the user's device.

Table 1: Key technical components.

Component	Implementation
Feature extraction	MobileNet v2 ($\alpha = 1$, 224×224 px)
Classifier	KNN ($k = 3$), Marcelle `knnClassifier`
Pretrained path	Raw TF.js `loadMobileNet`, top-20 ImageNet
Training storage	Marcelle `dataStore('localStorage')`
UI framework	Marcelle `wizard()`, reactive streams
LLM	Google Gemini API (model cascade)
Build tool	Vite (run: `npm run dev`)

4.2 Dataset

User-constructed training set. By default, **users construct the training set themselves** during the labeling step (Step 1). Each uploaded image is processed through MobileNet v2; the resulting 1280-dimensional feature vector and a 64×64 px JPEG thumbnail are stored in `localStorage` under the key `alcohol-training`. The store persists across page refreshes, so examples accumulate over multiple sessions.

Kaggle Alcohol Image Dataset (bootstrap). To lower the annotation barrier and let users start with a functional KNN immediately, the application ships with the *Alcohol Image Dataset*[1] from Kaggle. The dataset contains **141 JPEG photographs** of alcohol bottles captured in varied retail environments, accompanied by **141 Pascal VOC XML annotations** (one per image) that provide per-object bounding boxes; all objects share the single class label `alcohol_bottle`. One annotation file lacks an `<object>` element and is discarded, leaving **140 usable image–annotation pairs**.

Image resolutions range from 150×299 to 6 936×9 248. Bounding-box coverage varies widely: the annotated bottle region occupies between 3.9% and 95.7% of the image area (mean 35.9%), reflecting a mix of tightly cropped product shots and wide-angle shelf photos.

Preprocessing pipeline. A one-click *Load Kaggle Dataset* button on the labeling page triggers the following pipeline for each entry in a precomputed `manifest.json`:

1. **Crop.** The image is loaded in the browser, cropped to its bounding box, and rescaled to 224×224 px on a `<canvas>`.
2. **Auto-label.** A raw TensorFlow.js MobileNet v2 instance classifies the cropped region, returning the top-10 ImageNet predictions. A two-pass heuristic scans these predictions:
   - Generic container labels (*pop bottle*, *water bottle*, *vase*, etc. — 22 entries) are skipped.
   - The first prediction that matches an explicit ImageNet → alcohol-label mapping (21 entries, e.g. `wine bottle→Red Wine`, `beer bottle→Lager`) determines the label.
   - If no mapping matches, alcohol-keyword detection (25 keywords) assigns the fallback label `Liqueur`.

   Because every image in the dataset depicts an alcohol bottle, the heuristic never assigns *Non-alcoholic*.
3. **Feature extraction.** The Marcelle MobileNet v2 feature extractor produces a 1280-dimensional vector from the same 224×224 canvas.
4. **Storage.** The feature vector, auto-assigned label, and a 64×64 thumbnail are written to the `trainingSet` in `localStorage`.

Images are processed in batches of five to avoid blocking the browser event loop; a progress bar shows completion status. After loading, the user can review and manually correct any mislabeled examples in the dataset browser before training.

Pretrained classification path. For the pretrained (non-KNN) path, the application queries the full top-20 ImageNet probability distribution and filters it against the same curated list of 25 alcohol-related keywords. Non-alcohol ImageNet classes are collapsed into a single *Other / Non-beverage* bar in the confidence chart, keeping the visualization focused on the decision the user actually needs to make.

4.3 Label Vocabulary

The 26-class vocabulary was designed to balance coverage with usability:

[1] `https://www.kaggle.com/datasets/sripaadsrinivasan/alcohol-image-dataset`

- **Whiskies:** Scotch, Bourbon, Irish, Japanese
- **Wines:** Red, White, Rosé, Champagne, Prosecco, Port
- **Beers:** Lager, Craft Beer/IPA, Stout
- **Spirits:** Vodka, Gin, Tequila, Mezcal, Rum, Brandy/Cognac, Absinthe
- **Asian spirits:** Sake, Soju, Baijiu
- **Other:** Liqueur, Cider, **Non-alcoholic**

The *Non-alcoholic* class is treated as a first-class label so that false negatives surface as an explicit KNN prediction rather than the absence of a prediction.

4.4 State of Implementation

The full six-step wizard is functional end-to-end. Table 2 summarizes the implementation status of each planned feature.

Table 2: Feature implementation status.

Feature	Status
KNN training loop	✓ Complete
Pretrained MobileNet path	✓ Complete
Confidence meter (3-zone)	✓ Complete
FP/FN explainer + alert panels	✓ Complete
User decision gate (Accept/Flag)	✓ Complete
Active learning correction	✓ Complete
Taste profile (localStorage)	✓ Complete
Gemini enrichment + cascade	✓ Complete
Personalized LLM output tag	✓ Complete

Known limitations: the pretrained path relies on ImageNet classes, which do not include fine-grained brand distinctions (e.g., distinguishing a specific single-malt from a blended Scotch). The KNN path can achieve higher specificity once the user has labeled sufficient examples, but requires an initial annotation investment.

5 Proposed Evaluation

We propose a **mixed-methods lab study** with two conditions: (A) the full BottleWise interface and (B) a *baseline* variant in which confidence scores are hidden and the user must accept every prediction without the ability to flag or correct.

Participants. 12–16 non-native speakers with low-to-moderate familiarity with the local alcohol market, recruited from international student networks.

Task. Participants are shown 10 beverage images (selected to span high and low model confidence, and to include at least one false-negative case: an alcoholic drink the model predicts as Non-alcoholic). They use the app to identify each beverage and record their final decision.

Measures.

- **Decision accuracy** (primary): proportion of beverages correctly identified as alcoholic vs. non-alcoholic, compared between conditions. We are especially interested in *false-negative catch rate*: the proportion of the seeded false-negative cases that participants correctly identified as “uncertain / needs checking” rather than accepting the wrong prediction.

- **Appropriate trust** (Likert): “The AI is honest about when it is unsure” (5-point scale), adapted from the Explainability Satisfaction Scale [4].
- **Correction loop engagement**: number of corrections submitted per session, and the resulting change in KNN accuracy on the next identification within the same session.
- **Task completion time** and **step abandonment rate**.

Protocol. 5-minute onboarding, 20-minute task, and 10-minute interview probing mental models of uncertainty (“What did the red bar mean to you?”, “When would you trust the result?”).

Analysis. Decision accuracy and false-negative catch rate are compared between conditions using a Mann-Whitney U test (non-parametric, expected $n < 20$). Likert responses are compared with a Wilcoxon signed-rank test. Interview transcripts are thematically coded for (1) uncertainty comprehension, (2) trust calibration, and (3) perceived agency.

The study is designed to answer the core research question: *does exposing model uncertainty in plain language improve users’ ability to detect high-risk misclassifications without eroding trust in correct predictions?*

6 Conclusion

BottleWise shows that interactive ML principles, including uncertainty transparency, user agency, and human-in-the-loop correction, can be applied in a practical consumer app with real safety implications. The system is fully implemented in the browser with no server dependency, making it deployable without infrastructure. The central design contribution is the *uncertainty gate*: no LLM enrichment fires until the user has reviewed the model’s confidence and actively accepted the prediction, ensuring that the most dangerous failure mode (a false negative on alcohol content) is surfaced to the user before they act on it.

References

[1] Saleema Amershi, Maya Cakmak, W. Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[2] Jerry Alan Fails and Jr. Olsen, Dan R. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI ’03, pages 39–45, New York, NY, USA, 2003. Association for Computing Machinery.

[3] Jules Françoise, Baptiste Caramiaux, and Téo Sanchez. Marcelle: Composing interactive machine learning workflows and interfaces. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST ’21, pages 39–53, New York, NY, USA, 2021. Association for Computing Machinery.

[4] Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

[5] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. Updated January 26, 2010.