Multi-Dataset, Multi-Classifier Report

Group 29 (Bheda, Reyes, Velarde)

November 10, 2025

1 Introduction and Objectives

In this exercise, we conducted a comprehensive classification study using four diverse datasets, each originating from a different domain and presenting its own level of complexity and analytical challenges. Two of these datasets were selected by our team, allowing us to explore domains that offered interesting modeling opportunities, while the remaining two were assigned by the lecture team as part of the course requirements. This combination ensured a balance between guided exploration and independent experimentation, giving us the chance to apply our methods under both familiar and unfamiliar conditions.

The main objective of the project was to go through the entire supervised learning workflow in a systematic and reproducible manner. This involved every major stage of a typical machine-learning process — from the initial data acquisition and cleaning, through feature engineering and preprocessing, to model training, validation, and evaluation. By working with datasets of different sizes, levels of dimensionality, and degrees of class imbalance, we were able to observe how each of these factors affects model behavior and overall predictive performance.

To make our comparison meaningful, we implemented three distinct classifiers representing different methodological paradigms: a linear model (Logistic Regression), a tree-based model (XGBoost), and a margin-based model (Support Vector Machine). These algorithms differ significantly in their assumptions, flexibility, and computational demands, which made them ideal candidates for examining trade-offs between interpretability, accuracy, and runtime efficiency.

Our specific goals were to:

- Design reproducible experiments and analyze how different preprocessing choices(such as scaling, encoding, and imputation) affect classifier performance.
- Quantify both effectiveness (Precision, Recall, and F1-score) and efficiency (runtime) across all models and datasets.
- Examining which models generalize more consistently and how specific dataset properties influence predictive outcomes.

The remainder of this report introduces the datasets and preprocessing methods (Section 2), details the classifiers and experimental setup (Section 3), presents and compares results (Section 4), and concludes with key findings and reflections (Section 5).

2 Datasets

2.1 Overview

We have chosen our two datasets primarily to ensure a high level of diversity in the type and structure of the data we work with.

The first one, the Wine Review dataset (**URL OpenML Wine Review Dataset**), contains only a small number of numerical and categorical columns, but it also includes a free-text description column which contains a review for each wine. This textual component makes the dataset particularly interesting, as it opens the door to various text preprocessing and feature extraction techniques such as tokenization, TF-IDF vectorization, or sentiment analysis. Furthermore, this dataset represents a multiclass classification problem, which allows us to explore models capable of handling multiple categories simultaneously, and to evaluate their performance across several classes rather than a simple binary outcome.

In contrast, the AirBnB dataset (**URL OpenML AirBnB Dataset**) provides us with a much broader structure. It contains numerous attributes resulting in a high-dimensional multiclass task. This setup allows us to experiment with different feature selection and dimensionality-reduction methods, and to compare how various algorithms handle large and complex input spaces.

The Breast Cancer (**URL Kaggle Breast Cancer Dataset**) and Loan (**URL Kaggle Loans Dataset**) datasets, on the other hand, were provided to us as part of the exercise and therefore did not involve an active selection process from our side. Nonetheless, we have performed data profiling, exploration, and cleaning steps to understand their main characteristics.

A detailed summary of these datasets is presented in the following section.

2.2 Characteristics

In the following table we detail some of the characteristics of the datasets we have worked on.

Dataset	Samples	Feature Columns	Classes	Target
Kaggle Breast Cancer	285	30	2	`class`
Kaggle Loan	10.000	90	7	`grade`
Wine Reviews	84.123	5	29	`variety`
AirBnB	18.316	103	10	`price_label`

2.3 Preprocessing decisions

For each of the datasets, we specify the processing we have applied throughout the whole training process.

2.3.1 Kaggle Breast Cancer Dataset

- **Missing Values:** There are no missing or NULL values in this dataset. This ensures that the preprocessing pipeline can focus on scaling and validation rather than data imputation or categorical encoding.
- **Encoding:** Since all features are numerical, there was no need to apply categorical encoding. Each feature was treated as a continuous variable directly usable by all classifiers.
- **Scaling:** To ensure comparability among features and improve convergence for models such as Logistic Regression and Support Vector Machines, scaling was applied. Specifically, the scaling step was executed independently within each cross-validation fold to strictly avoid any form of data leakage. This procedure guarantees that the scaling parameters are learned exclusively from the training partition of each fold.
- **Outliers:** Visual inspection of the feature distributions and boxplots revealed the presence of several outliers. However, we chose not to remove them, as these extreme values may correspond to genuine measurements indicative of recurrent cases. Removing them could have led to a loss of valuable signal, particularly for the positive class, which tends to be more difficult to detect.

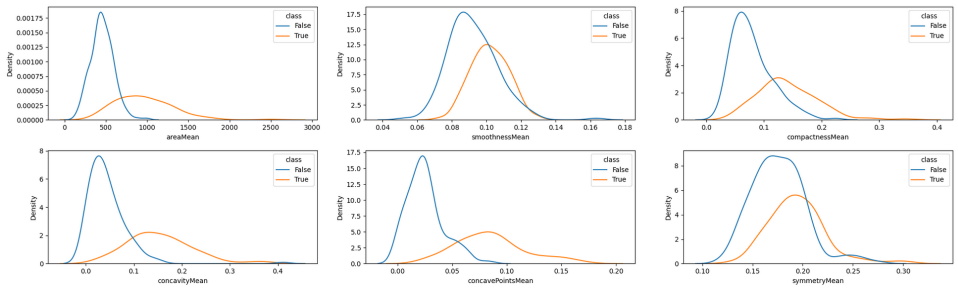

Figure 1: "Example: Cancer Recurrence EDA"

2.3.2 Kaggle Loans Dataset

- **Missing Values:** There are no missing or NULL values in this dataset. There was a `"NONE"` value in the categorical column **home_ownership** that was not present in the training set. This was resolved by changing the value to `"OTHER"`.
- **Encoding:** There were 13 categorical columns that had to be encoded as numerical values. We made an effort to capture meaningful relationships and not just use dummy variables for everything. This were the decisions we made:
- **Scaling:** Similar to the process used with the Breast Cancer Dataset, scaling was applied independently within each cross-validation fold. This results in better estimates for model performance with no data leakage during training.

Feature	Values	Encoding
term	`36` or `60`	Extracted decimal values
emp_length	From `<1` to `10+`	Extracted decimal values
home_ownership	Values like `RENT`	One-hot encoding
verification_status	Different verification levels	Numerical `0`, `1`, or `2`
loan_status	Different loan payment levels	Numerical from `0` to `5`
pymnt_plan	`y` or `n` values	Extracted binary values
purpose	Values like `CAR`	One-hot encoding
addr_state	Values like `CA` or `NY`	Frequency and label encoding
initial_list_status	Values `w` and `f`	Extracted binary values
application_type	Values `Individual` or `Joint`	Numerical `1` or `2`
hardship_flag	Values `Y` or `N`	Extracted binary values
disbursement_method	Values `Cash` or `DirectPay`	Numerical `-1` or `1`
debt_settlement_flag	Values `Y` ro `N`	Extracted binary values

- **Outliers:** We first checked the numerical columns to separate features that followed a normal distribution, the ones that showed considerable skewedness, and the multi-modal ones. Depending on this results, we used *Inter-Quartile Range* for skewed or *Isolation Forest* for multi-modal distributions to identify the outliers in a meaningful way. Finally, we performed a visual check of the distribution of the labels over the whole data next to the distribution of the labels for the outliers. Again, we chose not to remove any values.

2.3.3 Wine Reviews Dataset

- **Missing Values:** Several categorical columns contained missing entries, which were imputed by introducing an explicit "Unknown" category. For numerical columns, missing values were replaced by the mean of the corresponding feature. In both cases, an additional binary indicator column was created to flag whether the value had been imputed. This approach allows the models to capture potential patterns associated with missingness itself, rather than treating imputation as neutral.
- **Encoding:** All categorical variables were encoded using One-Hot Encoding, which expands the feature space but enables most machine-learning algorithms to handle categorical information effectively. To avoid data leakage between training and validation folds, encoding was performed independently within each cross-validation split. Furthermore, the **description** column, containing free-form text written by wine reviewers, was vectorized using the TF–IDF (Term Frequency–Inverse Document Frequency) representation. This step transforms text into a numerical form that captures both word importance and relative frequency across reviews.
- **Scaling:** Because the dataset includes numeric variables such as price and score, feature scaling was applied on each fold separately. This ensures that the scaling statistics (mean and variance) are derived only from the training data, preserving the integrity of the cross-validation process.
- **Outliers:** A number of observations exhibited unusually high or low price values, which we decided to retain rather than exclude. Our reasoning is that extreme

prices likely correspond to premium or very low-quality wines, and these cases are meaningful for distinguishing between different classes. Removing them would artificially narrow the range of variability, potentially harming the model's ability to learn meaningful distinctions among wine categories.

2.3.4 AirBnB Dataset

- **Missing Values:** The dataset exhibited substantial missingness across multiple columns. Review-related features (**review_scores**, **reviews_per_month**) showed approximately 25% missing values, while optional fields such as **security_deposit** (72%) and **cleaning_fee** (25%) had considerably higher rates. To preserve information about missingness patterns, we created binary indicator features (e.g., **has_security_deposit**, **no_reviews_yet**) before imputation. Numeric features were filled with either 0 (for **fees/deposits**) or median values (for **review_scores** and **property** characteristics), while binary features were imputed using mode values.

- **Encoding:**: This dataset required extensive categorical handling due to its heterogeneous structure. Boolean variables: 27 binary host verification fields (e.g., **host_verifications_email**, **host_verifications_phone**) were converted from string representations ('t'/'f') to numeric (0/1). Low-cardinality categoricals: One-hot encoding was applied to **room_type**, **bed_type**, **cancellation_policy**, and **state** (after normalization to consolidate spelling variations of Victoria, NSW, and Queensland). High-cardinality features: The **property_type** column (143 unique values) was encoded using frequency encoding to avoid excessive dimensionality explosion. Target encoding: The zipcode feature was target-encoded using the numeric price label with smoothing parameters to prevent overfitting. List-type features: Amenities (369 unique values) and host verifications were parsed from string representations and multi-label binarized, creating 369 and 24 binary columns respectively. Text features: Eleven text columns (**name**, **summary**, **description**, etc.) were combined and vectorized using sentence-transformers (all-MiniLM-L6-v2 model) to generate 384-dimensional embeddings per listing.

- **Scaling:**: StandardScaler was applied exclusively to numeric features within each cross-validation fold. Text embeddings, being pre-normalized by the transformer model, were not scaled. To prevent data leakage, the scaler was fitted only on training splits and then applied to both training and validation data.

- **Outliers:**: Outlier analysis revealed extreme values in several numeric columns, particularly in pricing-related features and review scores. However, these were retained as they represent legitimate market segments (e.g., luxury properties with ultra-high prices, or problematic listings with very low review scores). Removing such cases would have artificially constrained the model's ability to learn the full spectrum of price categories, especially the "ultra luxury" class.

3 Methods

3.1 Classifiers

We selected three classifiers that represent distinct learning paradigms — linear, margin-based, and ensemble-based — to ensure methodological diversity and to compare how different inductive biases perform across datasets with varied characteristics.

Logistic Regression (LR). A simple linear model that estimates class probabilities using a logistic (sigmoid) function. It serves as a strong baseline for well-behaved, linearly separable datasets and is computationally efficient. Its interpretability makes it suitable for understanding the influence of individual features, although it may struggle with complex nonlinear relationships.

Support Vector Machine (SVM). A margin-based classifier that seeks an optimal separating hyperplane between classes. We primarily used the linear kernel to maintain comparability with Logistic Regression and to handle high-dimensional spaces efficiently. SVMs are robust to outliers and can perform well even with limited samples, but are sensitive to feature scaling, making preprocessing critical.

Extreme Gradient Boosting (XGBoost). A tree-based ensemble model that builds multiple weak learners sequentially, optimizing a regularized objective function. XGBoost can model complex nonlinear relationships and typically handles feature interactions and missing values well. It is less sensitive to scaling but computationally more expensive, especially on large datasets.

3.2 Experimental design

For the experiments, we have selected a set of seeds which were used to reproduce results for every execution. Each seed gives us a different (but reproducible) split of the data, which is then measured on the validation dataset in order to get a measure of the mean performance of the model over these different data splits.

3.3 Metrics

Given Classification as the problem we are trying to solve, we selected Classification based metrics, in particular:

- Precision: Out of all the instances the model predicted as positive, how many were actually positive?
$$\frac{TP}{TP + FP}$$
- Recall: Out of all the actual positive instances, how many did the model correctly identify?
$$\frac{TP}{TP + FN}$$

- F1: The harmonic mean of precision and recall.

$$\frac{2 \times Precision \times Recall}{Precision + Recall}$$

Our main metric chosen is the F1 score, as this allows us to penalize strongly when either Recall or Precision is high at the expensive of the other metric. That is, for a High Precision and Low Recall, F1 penalizes the final score, and likewise for a High Recall and Low Precision.

4 Results

4.1 Efficacy

Below is a table detailing the mean performance for each model using the macroaverage of each metric for the final version of each model.

Dataset	Metric	Logistic Regression	Support Vector Machine	XGBoost
Breast Cancer	Precision	0.958*1	0.971*1	0.952*1
	Recall	0.962*1	0.970*1	0.955*1
	F1	0.960*1	0.971*1	0.953*1
Loans	Precision	0.877*3	0.629*3	0.902*3
	Recall	0.818*3	0.602*3	0.889*3
	F1	0.843*3	0.606*3	0.891*3
Wine Reviews	Precision	0.690	N/A*2	0.790
	Recall	0.700	N/A*2	0.758
	F1	0.694	N/A*2	0.770
AirBnB	Precision	0.311	0.321	0.363
	Recall	0.319	0.325	0.368
	F1	0.308	0.316	0.358

*1 As measured using the best cross-validation metrics, given that Kaggle holdout dataset is unlabeled.

*2 The model takes too long to train for even a single cv fold for 29 classes.

*3 Measured from 30% of the labeled data separated as test set.

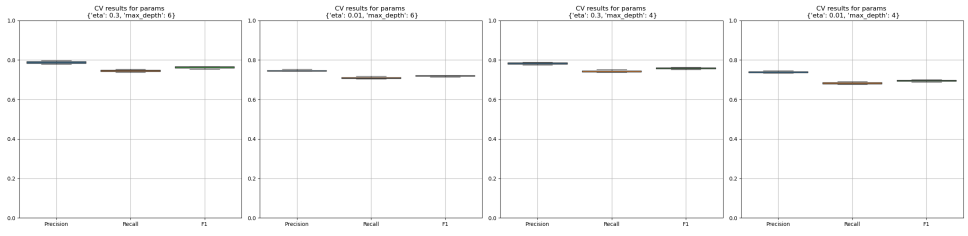

Figure 2: "Example: XGBoost Wine Classification CV Performance"

4.2 Efficiency

Below is a table detailing the mean execution time in seconds of the models over the different datasets. There could be some variability due to processing power of different machines. We can see that due to fact that the Wine Reviews dataset has so many classes, the SVM model has a much worse efficiency performance, as it must calculate the one-versus-others combinations for all the classes.

Dataset	Rows	Logistic Regression	Support Vector Machine	XGBoost
Breast Cancer	285	$1s$	$1s$	$1s$
Loans	10.000	$16s$	$13s$	8
Wine Reviews	84.123	$125s$	$> 1.800s$[*1]	$420s$
AirBnB	18.316	$11s$	$85s$	$26s$

[*1] The model takes too long to train for even a single cv fold for 29 classes.

5 Conclusions

5.1 General Conclusions

As the datasets grow more and more complex, we can see that XGBoost outperforms the more simpler models, however, for the very small and simple dataset that is the Cancer dataset, Logistic Regression holds up quite well, and in fact, gave us the best results.

5.2 Breast Cancer

For this dataset, we found that there was a slight imbalance in the classes, though not enough to deserve a special handling. When training a baseline model, we can notice we already have a decent performance, and all other techniques applied afterwards gave only marginal benefits, which nonetheless might be significant when applied to the domain of Cancer detection. Furthermore, as the amount of rows was so small, we found that there is a higher variability in the crossvalidation performance for this dataset, a small amount of data split into different parts can produce wildely different data distributions versus datasets with much more rows.

5.3 Loans

With the loans dataset, the several classes presented enough imbalance that the Support Vector Machine was initially having difficulties to complete. We used the `class_weight` parameter to compensate the imbalance. This dataset offered us the opportunity to engineer several features. We recognize that the relationships we decided to create for the numerical representation of some columns might influence the model in a negative way. This can be seen because a simple one-hot encoding for all features results in a better performance across all 3 methods. In the end, we decided to commit to our engineered features (since we had achieved a high rank on the Cancer competition already).

For the cross-validation we tried to eliminate any data leaking but we identified one way we leaked data during our process: the frequency and label encoding used for the address state feature. Overall, the process was very illuminating. Feature engineering can be a time-consuming endeavour that can result in data-leakage without providing better results than the use of dummy variables.

5.4 Wine Reviews

For this dataset, we observed that the distribution of the wine varieties is highly imbalanced. A few popular varieties appear very frequently, while many others occur only a few times. This imbalance poses a significant challenge for most classification algorithms, as models tend to overfit the dominant classes and fail to generalize properly to underrepresented ones. Moreover, the dataset contains a large number of unique labels—in our case, 29 different wine varieties—which further increases model complexity and makes it difficult for the classifier to learn meaningful decision boundaries across all classes.

One possible strategy to mitigate this issue would be to redefine the labeling scheme in order to simplify the prediction task. Instead of forcing the model to distinguish among all 29 categories, we could introduce a more aggregated representation. For instance, wine varieties that appear fewer than a certain number of times could be grouped into a single "*Other*" category. This would reduce noise and give the model a better chance of learning discriminative patterns from the most relevant classes without being overwhelmed by rare categories. Another potential approach would be to relabel each wine according to its broader color type—for example, "Red," "White," or "Other." This alternative formulation would convert the task into a three-class classification problem, likely resulting in more stable and interpretable results while still retaining an interesting level of complexity.

We also found that feature extraction played a crucial role in improving model performance. When the model relied solely on the structured numerical and categorical features, its predictive ability remained limited, suggesting that much of the distinguishing information lies within the free-text review descriptions. After incorporating TF–IDF encoding on the review text, we observed a clear and consistent improvement in classification accuracy. This demonstrates that the textual data provides valuable contextual cues about the flavor profile, aroma, and overall sensory experience of each wine, which in turn helps guide the model toward identifying the correct variety. In practical terms, this result highlights how the combination of structured and unstructured data can significantly enhance predictive performance and provide a richer understanding of the domain.

When considering a few sets of different parameters, we have seen that for both the Logistic Regression as well as the XGB Classifier model performed best with the default parameters.

Finally, we observed that for such a high amount of distinct classes, the SVM model is extremely unefficient, taking more than 30 minutes without being able to fit a single cross-validation loop.

5.5 AirBnB

Working with this dataset involved predicting price categories for 18,316 listings across 10 ordered classes from "very low" to "ultra luxury." The severe class imbalance (ratio of 0.030) and integration of diverse data types—numeric, categorical, text, and list-structured features—made this our most challenging task.

XGBoost significantly outperformed linear models, achieving 42.22% accuracy compared to Logistic Regression's 33.92% and SVM's 34.66%. This 8-percentage-point gap reveals that Airbnb pricing involves complex feature interactions that tree-based methods capture effectively while linear models cannot. F1-macro scores reflected the same pattern, with XGBoost reaching 0.358 versus 0.31 for linear classifiers.

Surprisingly, adding full 384-dimensional text embeddings decreased XGBoost's accuracy to 41.05% (-2.78%), though Logistic Regression improved slightly (+0.40%). This suggests text descriptions introduce noise when predicting discretized price bins. However, PCA compression to 50 dimensions (64% variance retained) recovered performance to 42.06% while reducing training time from 164 to 34 seconds—demonstrating that dimensionality reduction filters signal from noise effectively.

Five-fold cross-validation confirmed these findings with remarkable stability: XGBoost achieved 41.80% $\pm$ 0.34%, nearly identical to the 42.22% holdout result. Hyperparameter tuning via GridSearchCV yielded 41.61% accuracy, slightly below baseline, indicating default parameters were already near-optimal. The 42% ceiling reflects fundamental challenges: discretizing continuous prices into 10 bins obscures natural groupings, and temporal factors like seasonality aren't captured in our static features.

Computational efficiency varied dramatically: Logistic Regression trained in 9 seconds, XGBoost in 44 seconds, while SVM exceeded 30 minutes with full features. For production deployment, XGBoost offers the best accuracy-speed balance.

The key lesson: more features don't guarantee better performance. Adding 384 text dimensions hurt XGBoost's accuracy, emphasizing careful feature evaluation over indiscriminate inclusion. Tree-based methods' success confirms real-world pricing involves complex non-linear interactions, while holdout-CV consistency validates our experimental methodology.