John Knight
CSCI-6366-01
May 8, 2025

Final Project

The goal of this project was to predict binary coffee ratings based on a set of features using three classification models: decision tree, k-nearest neighbors (KNN), and multilayer perceptron (MLP). The model showing best performance on the training data was chosen to make predictions on the test set.

Preprocessing

There were 5 features (*roaster, roast, origin, 100g_USD, review*) and the binary response variable was *class*. Each of the features underwent some preprocessing. For the text column, *review*, Porter Stemmer from the *nltk* library was used to convert all words to their stems, and stop words were removed. Term Frequency-Inverse Document Frequency (TF-IDF) was used to increase the weight given to words with high frequency in a review, while penalizing words that appear in many reviews. Missing values for *roast* were imputed as the mode (Medium-Light) and a numeric variable was created with scores of 1, 2, 3, 4, and 5 mapping to Light, Medium-Light, Medium, Medium-Dark, and Dark. This was done with the aim of preserving the meaningful order from Light to Dark. The variable *roaster* had many categories with low frequency. This was a concern because one-hot encoding would convert these categories into binary variables with almost all 0s and very few 1s. Therefore, all categories of *roaster* with < 2.5% of the total rows (< 16) were classed as "Other". The same threshold was used for *origin*, but values were grouped geographically by continent giving "Other – Asia", "Other – Africa", and "Other – Americas". Finally, a histogram was created with *matplotlib* that showed the numerical feature *100g_USD* was heavily right-skewed, and therefore a log transformation was applied to better approximate a normal distribution.

Method and Implementation

The training data was split into an 80% cross-validation set, and a 20% validation set. Decision tree and KNN were built using *scikit-learn*, while the MLP was built using *pytorch*. For each of the three models, a pipeline was created to use grid search to try a range of hyperparameters (Table 1), chosen using some intuition, previous experience, and consultation of online resources.[4,5] Higher values of Min DF were tried for the KNN model to avoid the curse of dimensionality.[6] The best performing set of hyperparameters in cross-validation would then be used to make a prediction on the held out validation set, to see how it performed on unseen data. F1 score was used to assess performance in all cases. Whichever of the three models had the best F1 score on the validation set would be chosen as the final model to predict the test set.

Table 1. Hyperparameters tried in grid search for each method.

Method	Hyperparameters Tried in Grid Search
Decision Tree	Max Depth [2, 3, 5, 10, 15, None], Max DF [0.5, 0.7, 0.8, 0.9, 1.0], Min DF [1, 2, 5, 10]
KNN	K [2, 4, 6, 8, 10], Max DF [0.5, 0.7, 0.8, 0.9, 1.0], Min DF [1, 2, 5, 10, 20, 50]
MLP	HL [(50,), (100,), (200,), (100,5)], LR [1e-2, 1e-3, 1e-4], DR [0.2, 0.4, 0.6]

Max Depth = the maximum depth of the tree (root to leaf). Max DF = maximum document frequency, represented here as a proportion. Words appearing in a higher proportion of documents (i.e. reviews) than the parameter's value are not included in the model. Min DF = minimum document frequency, represented here as an integer. Words appearing in fewer than this number of documents are not included in the model. K = the number of nearest training samples used to assign class in KNN. HL = hidden layer, the shape of the MLP neural network. LR = learning rate, affecting convergence time and possibly under- or overfitting. DR = dropout rate, randomly setting a fraction of the neurons to zero which acts like an ensemble of smaller networks to help combat overfitting.

Additionally, the grid search above was performed using both the raw versions of the non-text features (*roast, roaster, origin, 100g_USD*) as well as the processed versions. Feature importance for KNN and MLP was assessed by Permutation Importance, which measures how much a model's performance decreases when a feature's values are randomly shuffled. For the decision tree, important features could be seen by inspecting the final decision tree itself.

Evaluation and Cross-Validation

Table 2. Cross-validation and validation set results using best parameters for each method.

Model	Best Parameters	CV Training	CV Test	Validation
DT – raw	Max Depth 10, Max DF 0.8, Min DF 1	0.9884	0.8994	0.8727
DT – proc.	Max Depth 3, Max DF 1.0, Min DF 1	0.9054	0.8963	0.8827
KNN – raw	k = 6, Max DF 1.0, Min DF 20	0.9002	0.8732	0.8712
KNN – proc.	k = 4, Max DF 1.0, Min DF 20	0.9095	0.8617	0.7974
MLP – raw	HL (100,50), LR 0.01, DR 0.6	1.0000	0.9554	0.9693
MLP – proc.	HL (50,), LR 0.01, DR 0.4	1.0000	0.9458	0.9586

As shown in Table 2, the decision tree and KNN models all had similar F1 scores of around 0.88 on the validation set, other than the KNN model with processed features which had a big drop between the CV test score and the validation score for some reason. All other models had similar CV test scores to validation scores, which is a good sign that overfitting was avoided during the CV process. The best performing method was MLP, which had F1 scores of 1.000 for CV training (i.e. perfectly fitting the data) but still as high as 0.9693 on the unseen validation set. The MLP with raw features slightly outperformed the one with processed features, suggesting it was better to keep the more granular information, and the MLP did a good job of not overfitting to it. The other best hyperparameters were two hidden layers of size 100 and 50, a learning rate of 0.01, and drop rate of 0.6.

Findings

In the superior MLP model, Permutation Importance revealed the most important feature to be the price of the coffee (*100g_USD*). As shown in Figure 1, the mean price of coffee with class 1 (“outstanding”) was around double the price of coffee with class 0 (“average”). The most important roaster was Dragonfly Coffee, and the most important word in the reviews was the stem “lilac” – both of these were associated with class 1. The most important feature that was associated with an outcome of class 0 was an origin of Burundi.

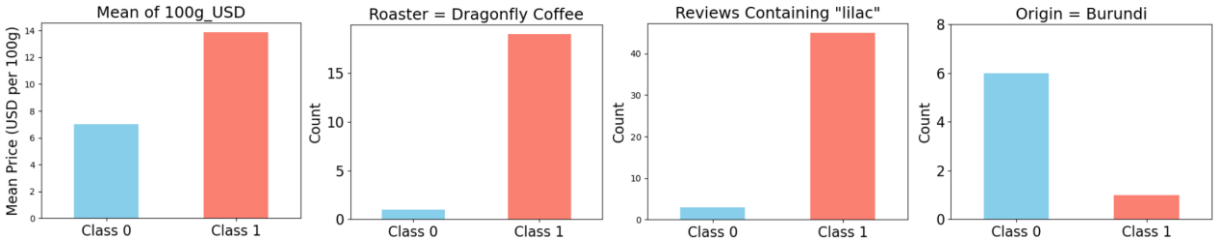

Figure 1. Comparison of class values for important features in final MLP model.

Interestingly, the decision tree produced with processed features (Figure 2) only used words from the reviews for its splits, and none of the other features. The first split was on the word stem “brisk”. Observations with a TF-IDF score greater than 0.142 (right branch) had 54 with class 0 (96.4%) and only two with class 1 (3.6%). So when it comes to coffee, “brisk” is bad. Likewise, the stem “gentl” was associated with class 0 – looking at the actual reviews, it seems both these adjectives are associated with the coffee’s acidity.

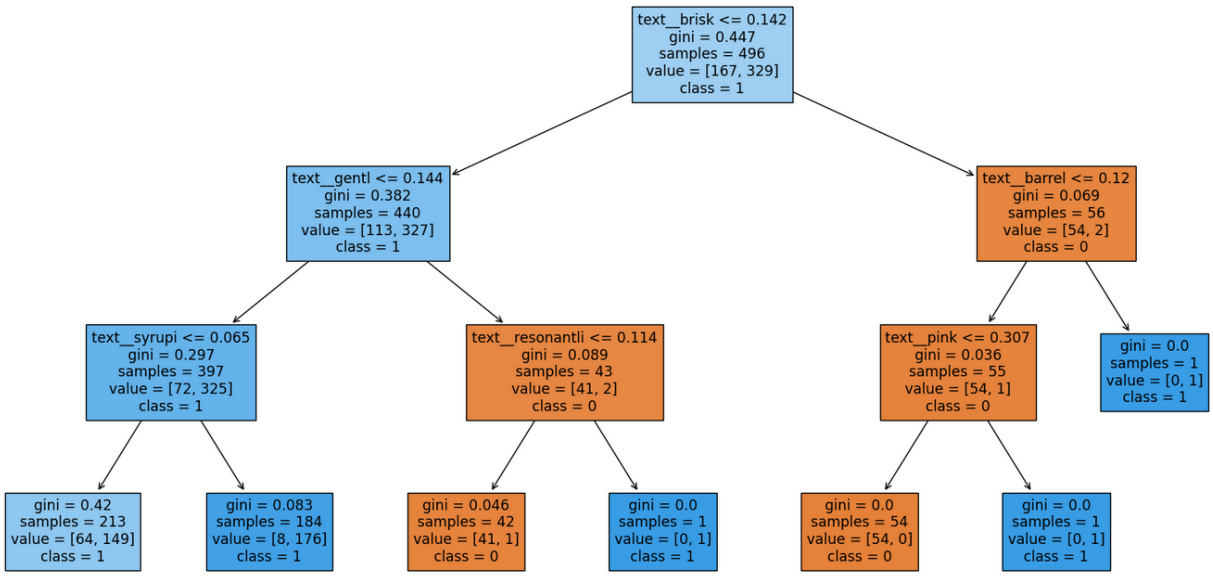

Figure 2. Decision tree of depth 3 trained using processed features.

Challenges

Some of the Python coding was challenging. In particular, it was tricky to set up a pipeline to perform grid search for the MLP model in pytorch – this ended up being a longer piece of code than I would have liked! I also ran into problems when trying to get Shapley values for the feature importance of MLP, as the *shap* library seemed to cause an error in my *numpy* installation, so I used Permutation Importance instead. I was able to resolve these difficulties with the help of a variety of online resources.[1-10] I was surprised that the pre-processed features did not perform better than the raw features which included one-hot encoding levels with as few as 1 true observation out of 620. However, it was no surprise that MLP turned out to be the best performing model.

Contributions

John Knight performed all work in this project.

References

1. https://codefinity.com/blog/A-Comprehensive-Guide-to-Text-Preprocessing-with-NLTK
2. https://towardsdatascience.com/how-to-turn-text-into-features-478b57632e99/
3. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
4. https://machinelearningmastery.com/building-multilayer-perceptron-models-in-pytorch/
5. https://www.youtube.com/watch?v=XNi5TPSxmZA
6. https://www.geeksforgeeks.org/curse-of-dimensionality-in-machine-learning/
7. https://stackoverflow.com/questions/71600683/reproducibility-issue-with-pytorch
8. https://www.youtube.com/watch?v=1SZocGaCAr8
9. https://chat.openai.com/chat
10. https://scikit-learn.org/stable/modules/permutation_importance.html