Task 5 Report

Jenny Sims
DATA 471
June 5th, 2025

1 Task

This report covers Task 5, a multiclass classification problem that predicts the style of beer from the specific ingredients used to brew the beer. The data provided are in the sparse file format, where each row is a triplet of numbers, with the first two numbers determining the index of where the third number goes in the matrix. The target values represent the style of beer.

2 Lead

Jenny Sims was primarily responsible for this task.

3 Methods

Based on the majority class of the dataset, the baseline training accuracy is 13.5%, calculated by the highest number of data points in one class divided by the total number of data points in the training target set. The baseline development accuracy is 13.6%, calculated the same way. We used these percentages to determine how successful our chosen models were. Initially, we parsed the data with SciKit-Learn's COO matrix, as it creates a sparse matrix to hand to our models.

We used three methods to estimate for this task: logistic regression, support vector machines specifically for classification (hence referred to as SVC), and XGBoost. These models were chosen to cover a range of linear and non-linear methods to see which would perform the best. Logistic regression was chosen to act as another baseline to compare against the other models, as it is a well-known linear model for multiclass classification. SVC was chosen as a starting point if non-linear methods would work better for this dataset. Finally, XGBoost was chosen for its high performance for a range of data, along with its ability to handle sparse data. Table 1 documents the reported accuracy of each of these three models without any hyperparameter tuning. For the initial hyperparameters, logistic regression iterated 1000 times, and SVC used the Radial Basis Function kernel (RBF). XGBoost had the number of estimators set to 2, the maximum depth set to 5, and the learning rate set to 0.01.

	Logistic Regression	SVC	XGBoost
Training Accuracy	0.269802	0.161832	0.275040
Development Accuracy	0.253248	0.159281	0.267685

Table 1: Accuracy of Initial Models

4 Submission Model Details

As XGBoost had the highest initial accuracy, we decided to continue with it for our hyperparameter tuning. XGBoost is a library in Python, with XGBClassifier being the actual model that we used. The default parameters include the number of estimators, maximum depth, learning rate, and the objective. We tuned the number of estimators, maximum depth, and learning rate throughout the tests, but kept the objective the same as "multi:softmax" as it is a multiclass classification task. We did not use any pre-processing because it had a minimal effect on performance.

5 Results

XGBoost performed well against our baselines, exceeding both the majority vote percentage and logistic regression training and development accuracy. We noticed that shallow trees, while they did not overfit, tended to underfit and not capture the complexity of the dataset, while deeper trees gave excellent training accuracy at the cost of overfitting. This was due to the high variance introduced with the higher depth. We used a ratio between the training and development accuracies to determine how much a certain test overfit. Our highest performing test was 1000 estimators at a depth of 50, with 99.4% train accuracy and 42.7% development accuracy, but this test did overfit.

Number of Estimators	Maximum Depth	Learning Rate	Train Accuracy	Dev Accuracy	Ratio of Train and Dev Accuracy
2	2	0.01	0.230108	0.227919	0.9904870756
2	5	0.01	0.27504	0.267685	0.9732584351
5	5	0.01	0.279224	0.270266	0.9679182305
5	5	0.05	0.293091	0.279759	0.9545124211
10	10	0.05	0.359218	0.316287	0.8804876148
200	10	0.05	0.552055	0.381294	0.6906811821
200	200	0.05	0.992509	0.423816	0.4270147676
1000	50	0.1	0.994258	0.426747	0.4292115326
1000	7	0.05	0.617903	0.408067	0.660406245
1500	7	0.05	0.661047	0.415285	0.6282231067
1500	7	0.1	0.745476	0.422416	0.5666393016
5000	7	0.05	0.812101	0.427228	0.5260774214

Table 2: Hyperparameter Tuning

6 Distribution of Work

Jenny Sims did the implementation, debugging, experimentation, result analysis, and report writing. Andy Ngo contributed with the formatting of his paper, and additional advice and clarifications were provided by all group members through multiple discussions.