Machine Learning Final Project

### Overview

Our team modeled all three targets using Python and SciKitLearn. Preprocessing included exploring the data and one-hot encoding the only categorical feature (V45). We passed each downsampled training set of 10,000 observations to a model selection process to determine which model types merited more specific consideration. Once the models of interest were uniquely tuned, we retrained the strongest-performing models on the full training dataset and applied them to the validation data. This resulted in the final models listed below with their goodness-of-fit metric:

| **Table 1: Selected Model Performance** | | |
| --- | --- | --- |
| **Target** | **Model** | **Performance of Validation** |
| Continuous | GBM | MAE: 0.2147 |
| Binary 1 | Naive Bayes | AUC: 0.6999 |
| Binary 2 | Naive Bayes | AUC: 0.6782 |

### Model Selection

We used 25-fold cross-validation to test multiple modeling techniques at once on the downsampled training data to gain an initial sense of which models performed well. In this first stage, we left all hyperparameters at their default values. Scores were evaluated using MAE for the continuous target, and AUC for the binary targets.

### Continuous Target - Model Development

The random forest model and gradient boosting machine best predicted the continuous target variable initially.

We employed a grid search to tune hyperparameters for the random forest model on our downsampled data set (Table 2, Appendix). The resulting MAE on the entire validation data set was 0.2366.

We tuned the gradient boosting machine using an alternative methodology. Each individual hyperparameter was uniquely tuned, and once it had reached its optimal level, it was held constant while the next parameter was optimized, and so on (Table 3, Appendix). The resulting MAE on the validation set was 0.2147. Because the GBM outperformed the random forest model, we used it to make our test-set predictions for submission.

### Binary Targets - Model Development

We selected models to predict both binary variables with an identical process. Independent, downsampled grid searches, scored by AUC, were used to identify random forest, gradient boosting, SVM, and Naive Bayes models as the most likely to produce sufficient predictions.

Table 4 (Appendix) displays the calculated validation AUC for each combination of model and binary target. Naive Bayes produced the strongest models for both binary targets by a large margin.

### Results & Recommendations

We produced models for each target variable exceeding baseline estimates. Gradient boosting proved to be the strongest at predicting the continuous variable, while Naive Bayes excelled at classifying both binary targets.

While pleased with our results, we believe performance could be improved through the following recommendations:

1. During initial model selection, utilize a cross-validation process that also implements surface-level hyperparameter tuning. Some models, like random forests and gradient boosting, excelled with their default values compared to other models. Should ]others have their own hyperparameters tuned, even just a tad, they might have been much more competitive.
2. Develop visualizations to inspect tuning parameters for signs of overfitting during grid searches. While letting the algorithm only output the strongest hyperparameter combination may be just as effective, such visuals would garner more confidence in the process and establish combinations where parameters may be overfitting.
3. Use computing resources to tune additional models rather than tuning hyperparameters which are unlikely to change a model’s fit. Certain modeling processes may be more adept than the ones we chose at handling our data. It could have been more worthwhile to look into more model-type/hyperparameter combinations in the initial stage to find a stronger predictor.

We thoroughly enjoyed taking part in the competition and getting a taste for how machine learning techniques are utilized in a project setting. We look forward to the results of the competition, and applying what we learned in future projects.

### Appendix

| **Table 2: Final Hyperparameters for Random Forest Model** | |  |
| --- | --- | --- |
| **Hyperparameter** | **Hyperparameter Value** | **Model MAE** |
| Number of Trees | 45 | 0.2366 |
| Maximum Features | 48 |
| Maximum Depth | 10 |

| **Table 3: Final Hyperparameters for Gradient Boosting Machine** | |  |
| --- | --- | --- |
| **Hyperparameter** | **Value** | **Model MAE** |
| Layers | 300 | 0.2147 |
| Subsample Value | 0.8 |
| Maximum Features | 42 |
| Learning Rate | 0.001 |
| Maximum Tree Depth | 10 |

| **Table 4: Model AUC by Binary Target** | | | | | |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Hyperparameter** | **Binary Target 1 Hyperparameters** | **Binary Target 2 Hyperparameters** | **Binary Target 1 AUC** | **Binary Target 2 AUC** |
| Random Forest | Number of trees | 100 | 175 | 0.5457 | 0.5571 |
| Maximum features per tree | ‘sqrt’ | ‘sqrt’ |
| Maximum Depth per Tree | 5 | 15 |
| Gradient Boosting | Learning Rate | 0.1 | 0.1 | 0.5427 | 0.5659 |
| Maximum Depth | 3 | 3 |
| Number of Trees | 25 | 25 |
| SVM | Regularization Parameter, | 10 | 25 | 0.5427 | 0.5659 |
| Kernel Type | ‘rbf’ | ‘rbf’ |
| Naive Bayes | Variable Smoothing Parameter | 0.001 | 1e-05 | **0.6999** | **0.6782** |