WEEK 8

For this week, we developed an XGBoost model to predict customer churn, utilizing its ability to handle large datasets, deal with missing data, and process various types of data, including numerical and categorical.

The modeling approach can be complex, as XGBoost is a gradient boosting algorithm that uses an ensemble of decision trees. Each decision tree is built sequentially to correct the errors of the previous tree. The model's complexity is influenced by several hyperparameters, such as the number of trees, learning rate, maximum depth of the tress, and subsample rate. Generally, increasing the number of trees or the depth of each tree increases the model's complexity. However, XGBoost has several built-in regularization techniques that help prevent overfitting.

To evaluate the model's performance, we selected performance metrics such as accuracy, precision, recall and F1 score. As our dataset was imbalance, with class 0 being much more frequent than class 1, we paid more attention to precision and recall, as accuracy could be misleading. We split the dataset into a training and validation set and calculated the performance metrics for both sets. We evaluated three variations of the model: the base model with default parameters, the tuned model with decreased learning rate and max_depth, and the model with increased min_child_weight and further decreased max_depth. The base model was our starting point, using default hyperparameters. The second model decreased both the learning rate and max_dpeth. From the table below, we could see that while the first two models had high accuracy, precision, recall and f1 score on the training dataset, they did not perform as well on the validation set. Model 2 even reached 99% for accuracy, precision and recall on the training dataset, indicating overfitting, as its accuracy on the validation set was much lower at 85%. This suggests that the model was too complex, and decreasing the learning rate and max_depth did not help to generalize it better to new data.

In contrast, the third model with further decreased max_depth and increased min_child_weight, achieved lower performance on the training set, with a precision of 85% and a recall of 74%. However, it achieved 82% precision on the validation set, indicating that it was better able to

generalize to new data. Before building the third model, we used grid search to find the optimal parameters for max_depth and min_child_weight. After getting the results, we built the third model with both max_depth and min_chile_weight set to 3. Of the three models evaluated, the third model achieved the highest accuracy and precision on the validation set. However, its recall value of 72% was not high. It is worth noting that all three models had a recall of 72%, which may not be ideal for this particular scenario, where accurately identifying all churn cases is crucial. Achieving high recall is desirable because it means that the bank can take appropriate actions to retain customers who are likely to churn, ultimately reducing customer attrition rates and improving overall customer satisfaction.

Overall, the XGBoost model was effective in predicting customer churn, but further tuning was needed to achieve optimal performance. The second model, with high accuracy on the training set, was overfitting the data and did not generalize well to new data. The third model, which achieved the highest accuracy and precision on the validation set, showed some improvement in generalization, but still struggled with recall. Further tuning, such as optimizing the subsample rate and number of trees, can also improve the model's performance.

Model	Train Accuracy	Train Precision	Train Recall	Train F1-Score	Validation Accuracy	Validation Precision	Validation Recall	Validation F1-Score
Default hyperparameters	0.965000	0.969479	0.922553	0.943710	0.860000	0.804441	0.727942	0.755506
Decreased learning_rate and max_depth	0.998437	0.999018	0.996203	0.997603	0.851250	0.783214	0.721307	0.744574
More Decreased in max_depth	0.877344	0.854758	0.742479	0.779522	0.863125	0.820092	0.720815	0.753502