Project 3: Classification Modeling Report

By Sibi Seenivasan (66176266)

I completed the project in a two-stage approach: first, strictly cleaning and preparing the raw data to ensure quality inputs, and second, establishing a baseline model before experimenting with advanced ensemble techniques to maximize predictive performance.

Data Cleaning and Preparation

Before modeling, I thoroughly examined the dataset to handle inconsistencies and prepare features for ingestion.

- **Data Examination & Handling:** I checked for missing values and handled them appropriately (imputation) to prevent data leakage or errors during training. I also identified constant features that offered no predictive value and removed them.
- **Feature Engineering:** I utilized discretization to bin specific continuous variables, helping to capture non-linear relationships.
- **Visualizations:** I used correlation heatmaps to understand the relationships between features and the target `y`, which helped inform my feature selection later.
- **Preprocessing:** I performed a `train_test_split` to reserve unseen data for the final evaluation.

Part A - Logistic Regression Model (The Baseline)

Model Explanation:

I started by establishing a baseline using Logistic Regression. This is a statistical model that uses a logistic function to model a binary dependent variable. It predicts the probability that a given input point belongs to a certain class (0 or 1).

- **Advantages:** It is highly interpretable, computationally efficient, and less prone to overfitting on small datasets.
- **Disadvantages:** It assumes a linear relationship between the independent variables and the \`log-odds\` of the target, which means it struggles to capture complex, non-linear patterns.

Methodology:

- **Feature Selection:** I utilized RFE (Recursive Feature Elimination) to identify the most impactful features, narrowing the scope to the signals that mattered most.
- **Tuning:** I utilized Cross-Validation to ensure the model's stability across different subsets of the data.

Results:

The Logistic Regression model performed strictly "okay" but left room for improvement.

- **Confusion Matrix:** Showed a decent balance but misclassified a noticeable portion of the positive class.
- **Metrics:** Yielded an Accuracy of **0.88** and an ROC-AUC of **0.94** on the test set.

Part B - Non-Logistic Model (Ensemble Tree-based)

Model Selection:

I then moved on to advanced models to see if I could improve performance. I ran a comparison between Random Forest, Gradient Boosting, XGBoost, and LightGBM. From the validation results, the XGBoost (Extreme Gradient Boosting) model commanded a lead over the others, so I selected it as my champion model.

- **Advantages:** XGBoost is highly efficient, handles missing values internally, and uses ensemble learning (combining many weak prediction trees) to capture very complex, non-linear patterns that Logistic Regression misses.
- **Disadvantages:** It is more of a "black box" (harder to interpret than Logistic Regression) and requires careful tuning of many hyperparameters to prevent overfitting.

Methodology:

- **Hyperparameter Tuning:** I implemented hyperparameter tuning (using `RandomizedSearchCV`) on the XGBoost model with K-Fold cross-validation. This optimized parameters like learning rate and max depth.
- **Feature Importance:** Post-modeling, I analyzed feature importance. Features `x14`, `x26`, and `x9` dominated the model, contributing approximately **58%** of the total importance.

Results:

The tuned XGBoost model demonstrated superior predictive capability.

- **Metrics:** It achieved an Accuracy of **0.97** and an ROC-AUC of almost **1.00 (0.999)**.
- **Challenges:** The main challenge was ensuring the model didn't overfit given the high AUC, but the cross-validation scores remained consistent with the test scores.

Discussions

Comparison of Approaches

When comparing the two approaches, the **XGBoost model significantly outperformed the Logistic Regression baseline**. While the Logistic model provided a strong starting point (88% accuracy), it hit a ceiling because it couldn't capture the complex interactions between variables. XGBoost, by learning from the errors of previous trees in its ensemble, was able to close that gap, pushing accuracy to 97%.

Performance Table

Metric	Model 1 - Logistic Regression	Model 2 - XGBoost (Tuned)
AUC for Training	~0.94	~1.00
AUC for Validation	~0.93	~0.99
AUC for Testing	**0.94**	**~1.00**
Accuracy	0.88	0.97

Business Explanation

If I were explaining this to a business executive, I would describe it this way:

We can think of the **Logistic Regression** model as a strict checklist. It looks at the data and makes a decision based on a fixed set of rules: if 'X' is high and 'Y' is low, it predicts 'Yes.' It's reliable and easy to understand, getting it right about 88% of the time.
The **XGBoost** model, however, acts more like a committee of experts. Instead of one checklist, it builds hundreds of small decision trees, where each new tree specifically focuses on correcting the mistakes the previous ones made. It learns the subtle nuances and 'grey areas' that the checklist misses.
Because of this ability to learn from its own errors, the XGBoost model is accurate 97% of the time. While the checklist is good, the 'committee' approach is far superior for catching the difficult cases we care about.

Conclusion

In conclusion, while the baseline was useful for understanding the data structure, the **tuned XGBoost model** is the robust choice for deployment, offering the highest reliability and precision for this dataset.