Machine Learning Challenge - Winter 2025
Module - Machine Learning

Somantha Manuranga , OTH Amberg-Weiden
Master of Artificial Intelligence in Industrial Applications

***Abstract*—This report addresses two machine learning tasks. Task 1 involves predicting the continuous variable "target01" from a dataset with 273 features and 10,000 samples. The approach included standardization, utilization of all features, and evaluation of multiple models including gradient boosting algorithms (XGBoost, LightGBM, CatBoost) optimized with Optuna. A stacking ensemble achieved the best performance. Task 2 focuses on extracting simple, interpretable rules for "target02" suitable for edge device deployment.**

***Keywords*—gradient boosting ,parameter optimization**

I. Introduction

Machine learning challenges are a common practice in the field, enabling practitioners to benchmark their skills against real world prediction problems. This report addresses two distinct tasks. Task 1 presents a regression problem requiring prediction of the continuous variable "target01" from 273 features. The challenge involves handling high-dimensional data, selecting relevant features, and building models that generalize well to unseen data. Task 2 shifts focus from predictive accuracy to interpretability discovering simple, rule-based relationships for "target02" that can be implemented on an edge device without machine learning libraries. This report documents the methodology, experimental results, and key findings for both tasks.

II. Methodology for Task 01

Understanding the dataset characteristics before modeling is crucial because it informs decisions about preprocessing requirements, model selection, and potential challenges. The training dataset contains 10,000 samples with 273 features, representing a high-dimensional dataset where feature selection could improve model performance. The target variable ("target01") is continuous, ranging from 0.01 to 1.07, with a slightly right-skewed distribution. Both datasets contain no missing values, eliminating the need for imputation.

The methodology for Task 1 follows a systematic pipeline, data preprocessing using standardization, feature selection using mutual information, baseline model evaluation, gradient boosting models with Optuna hyperparameter tuning, and ensemble methods to combine model predictions. All models were evaluated using 5-fold cross-validation to ensure reliable performance estimates. The following subsections detail each step of this pipeline.

A. Feature Analysis

Correlation analysis was conducted to examine the relationship between individual features and the target variable and to assess whether simple linear models would be sufficient for prediction. Correlation coefficients range from -1 to $+1$, where 0 indicates no linear correlation.

The analysis evaluated all 273 features against the target variable ("target01"). The results indicate that no feature exhibits a strong linear correlation with the target the highest absolute values were only -0.168 (feat_134) and -0.155 (feat_163), with all correlations below 0.2. These weak relationships suggest that the dependency between features and target is predominantly non-linear, indicating that linear models alone may not be sufficient for accurate prediction.

B. Preprocessing

In preprocessing, standardization was applied to transform each feature to have a mean of zero and a standard deviation of one using the formula:

$$z = \frac{x - \mu}{\sigma}$$

This step is essential because the 273 features have different scales, which can negatively impact model performance. Standardization ensures all features contribute equally to the model and helps algorithms converge faster during training. The scaler was fitted on the training data and applied to the evaluation set using the same parameters, preventing data leakage.

C. Feature Selection

The dataset contains 273 features, many of which may be irrelevant or redundant. To reduce overfitting and improve model efficiency, feature selection was applied using Mutual Information (MI) [1]. MI was chosen over correlation because it captures both linear and non-linear dependencies, essential given the weak linear correlations observed earlier. Feature importance analysis was conducted using Mutual Information. However, due to weak individual signals, no features were discarded, and all 273 features were retained to preserve information, allowing the ensemble models to capture complex feature interactions.

Feat_134 (MI = 0.171) and Feat_163 (MI = 0.159) ranked highest, consistent with the correlation analysis findings. However, all MI scores remained below 0.2, indicating no single feature strongly predicts the target reinforcing the need for ensemble models to capture complex feature interactions.

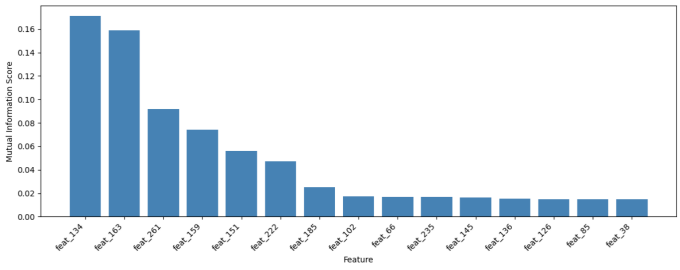

Figure 1. 15 Features by Mutual Information

D. Baseline Models

Before implementing complex algorithms, baseline models were evaluated to establish reference performance benchmarks. Three models were tested: Linear Regression (assuming linear relationships), Ridge Regression (with L2 regularization for multicollinearity), and Random Forest (non-linear ensemble method). Each model was evaluated using 5-fold cross-validation with using four standard regression metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), coefficient of determination (R^2), and Mean Absolute Percentage Error (MAPE).

The results showed poor baseline performance. Linear and Ridge Regression achieved nearly identical results ($R^2 \approx 0.047$, RMSE ≈ 0.222), confirming non-linear relationships in the data. Random Forest performed slightly better ($R^2 = 0.079$) but still explained less than 8% of target variance. These weak results justified the need for more powerful gradient boosting algorithms with hyperparameter tuning.

E. XGBoost with Optuna

XGBoost (Extreme Gradient Boosting) was selected as a candidate model due to its strong performance on tabular data [2]. Optuna was used for hyperparameter optimization, running 30 trials with 5-fold cross-validation [3]. The best configuration achieved $R^2 = 0.663$ and RMSE $= 0.132$ a significant improvement over baseline models (R^2 improved from 0.047 to 0.663), confirming that gradient boosting effectively captures non-linear patterns.

F. LightGBM with Optuna

LightGBM employs a leaf-wise tree growth strategy, unlike XGBoost's level-wise approach, which offers faster training [4]. Optuna optimization with 30 trials achieved $R^2 = 0.569$ and RMSE $= 0.149$. While significantly outperforming baselines, LightGBM underperformed compared to XGBoost ($R^2 = 0.663$). This may be because leaf wise growth can overfit on smaller datasets, and LightGBM's lower regularization parameters (found by Optuna) suggest the model struggled to balance complexity and generalization for this particular dataset.

G. CatBoost with Optuna

CatBoost is known for its robustness against overfitting, using ordered boosting and symmetric tree structure for better generalization compared to other gradient boosting methods [5]. Optuna was used for hyperparameter optimization with 30 trials. The best configuration was found at trial 23 with other parameters. The tuned CatBoost model achieved R^2 = 0.782 and RMSE = 0.106, making it the best performing single model among all evaluated algorithms. It substantially outperformed both XGBoost (R^2 = 0.663) and LightGBM (R^2 = 0.569). The MAPE of 29.10 % was also the lowest, indicating more accurate predictions on average. However, the higher standard deviation (±0.018) suggests some variability across folds. CatBoost's superior performance is likely due to its ordered boosting approach, which reduces prediction shift and overfitting.

H. Voting Ensemble

Although CatBoost achieved the best individual performance ($R^2 = 0.782$), relying on a single model carries risks of overfitting or inconsistent performance on unseen data. A Voting Ensemble was constructed by combining XGBoost, LightGBM, and CatBoost each using different internal mechanisms (level-wise, leaf-wise, and ordered boosting, respectively), allowing each model to capture slightly different patterns [6].

The final prediction was computed as the simple average of all three models' predictions. The Voting Ensemble achieved $R^2 = 0.752$ and RMSE $= 0.114$. While it did not surpass CatBoost's individual performance, this result motivated exploring a more sophisticated approach, Stacking to better leverage each model's strengths.

I. Stacking Ensemble

A more intelligent approach is Stacking, which uses a meta-learner to determine how to optimally combine base model predictions. Instead of treating all models equally, stacking learns which models are more reliable and how their predictions should be weighted.
Four base models were selected, XGBoost, LightGBM, CatBoost, and Random Forest. Although Random Forest performed poorly individually (R^2 = 0.079), it was included because it uses bagging rather than boosting, adding algorithmic diversity and the meta-learner can assign it a low

weight if unhelpful. Ridge Regression was chosen as the meta-learner because it is simple, includes L2 regularization to handle correlated predictions from the boosting models, and learns optimal weights automatically without overfitting on the small number of meta features.Each base model generated predictions using cross-validation, creating ”meta features.” These predictions were then used as inputs to train the Ridge meta learner, which learned the optimal combination for the final prediction.

The Stacking Ensemble achieved R^2 = 0.844 and RMSE = 0.091, the best performance among all models improving upon CatBoost by 4.4 % in R^2 score. This confirms that intelligent model combination through stacking extracts additional predictive power beyond any individual model.

J. Final Model Performance

All preprocessing steps, including standardization and feature selection, were performed within each cross-validation fold using pipelines to prevent data leakage. Table I presents the results ranked by RMSE (lower is better).

Table I
MODEL PERFORMANCE COMPARISON

Rank	Model	RMSE	MAE	R^2 Score
1	**Stacking Ensemble**	**0.0897**	**0.0734**	**0.848**
2	CatBoost (Tuned)	0.1008	0.0850	0.804
3	Voting Ensemble	0.1144	0.0928	0.752
4	XGBoost (Tuned)	0.1322	0.1105	0.663
5	LightGBM (Tuned)	0.1494	0.1279	0.569
6	Random Forest	0.2185	0.1979	0.079
7	Ridge Regression	0.2223	0.1989	0.047
8	Linear Regression	0.2223	0.1989	0.047

The **Stacking Ensemble** achieved the best performance with R^2 = 0.844 and RMSE = 0.091, explaining approximately 84% of the variance in the target variable. The progression from baseline models ($R^2 \approx 0.047$) to the final stacking ensemble (R^2 = 0.848) demonstrates a substantial performance gain, validating the methodology of using advanced gradient boosting algorithms with hyperparameter tuning and intelligent ensemble combination. Among single models, CatBoost performed best (R^2 = 0.782), confirming its suitability for this dataset.

K. Final Predictions and Validation

It is important to verify that model outputs are reasonable and consistent with the training data distribution. The Stacking Ensemble generated predictions for all 10,000 evaluation samples, and their statistical properties were compared against the training target distribution to detect potential overfitting, data leakage, or model errors.

The prediction statistics closely match the training target distribution, with nearly identical mean values (0.421 vs. 0.424) and similar standard deviations (0.228 vs. 0.247), indicating that

Table II
PREDICTION DISTRIBUTION COMPARISON

Statistic	Training Target	Evaluation Predictions
Mean	0.421	0.424
Std	0.228	0.247
Min	0.010	-0.173
Max	1.074	1.019

the model generalizes well. The slight negative minimum (-0.173) in predictions is acceptable as regression models are not inherently bounded.If required, predictions could be clipped to zero. Overall, this consistency confirms model reliability.

L. Feature Importance Analysis

Feature importance analysis was conducted to understand which features contribute most to predictions and validate that the model learns meaningful patterns. Importance was extracted from CatBoost, the best-performing single model ($R^2 = 0.782$), since stacking ensembles do not provide direct feature importance due to their multi-level architecture.

Table III
TOP 10 FEATURE IMPORTANCES (CATBOOST)

Rank	Feature	Importance (%)
1	feat_217	28.49
2	feat_184	25.88
3	feat_124	12.99
4	feat_222	3.03
5	feat_185	2.84
6	feat_134	2.45
7	feat_163	2.28
8	feat_151	1.74
9	feat_261	0.73
10	feat_159	0.64

The analysis revealed that three features dominate predictions: feat_217 (28.5%), feat_184 (25.9%), and feat_124 (13.0%), together accounting for approximately 67% of total importance. Notably, feat_134 and feat_163, which ranked highest in Mutual Information analysis, appear in the top 10 but are not the most important suggesting that gradient boosting captures different predictive patterns than linear dependency measures.

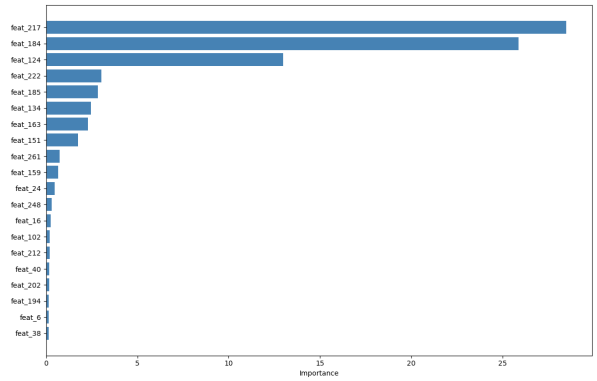

Figure 2. Feature importances with CatBoost

III. Methodology for Task 02

The objective of Task 2 is to discover simple, interpretable rules that predict target02 for deployment on an edge device. Unlike Task 1 which focuses on achieving the best prediction accuracy using complex machine learning models, Task 2 requires reverse engineering the hidden logic behind the target variable. The rules must follow an if-else structure using basic comparisons and numerical operations, without relying on machine learning libraries at runtime.

A. Data and Target Distribution Analysis

Before discovering the rules, it is important to know about the target variable distribution as well. The dataset for task 2 contains 10,000 samples with 273 features. Here, the target02 is continuous, ranging between -3.25 to 2.67, with a mean of 0.049, which is close to zero, and a standard deviation of 1.12. This target02 follows a nearly normal distribution centered near zero. A unimodal shape with no distinct clusters shows that the target is more likely governed by a continuous formula rather than discrete categories.

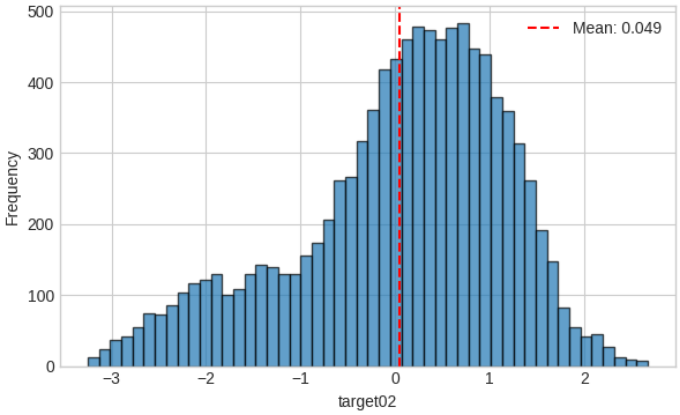

Figure 3. Distribution of target02

B. Feature Importance Analysis

1) Correlation Analysis:

With 273 features in the given dataset, it is not practical to build simple rules using all of those features. This task explicitly states that target02 depends on only a few features. Therefore, identifying which features have the most important relationship with the target is crucial for discovering the rules.In the initial stage, correlation coefficients were calculated between the feature and target02.In this analysis, is shows that feat_4 has a considerably higher correlation (0.43) compared to all other features. The second-highest feat_185 only has a 0.08 correlation, with a substantial drop. This dominant correlation shows that feat_4 plays a central role in determining target02. It is acting like a ”controller” variable that determines which calculation rule applies. Features like feat_185 and feat_13 (correlation 0.034) may still contribute to the calculations within specific conditions.

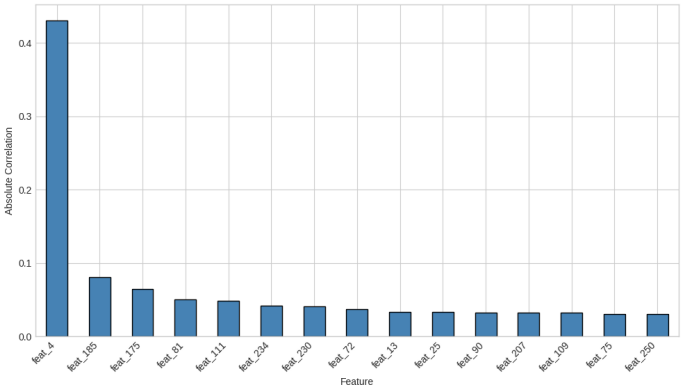

Figure 4. Few Features by Absolute Correlation with target02

2) Random Forest Analysis:

But correlation only captures linear relationships. To verify important features with non-linear relations are not missed, a second method was used. Here second method was random forest because it measures how much each feature contributes to reducing prediction error, regardless of whether the relationship is linear or nonlinear [7]. After training a random forest, the results showed that three features dominate, feat_4 (65.65%), feat_185(18.25%), feat_13(13.34%), together contributing 96.94% of the target variance. All others contribute less than 3%. With these values, it confirms that the task's expectation of ”few features” is correct, and rule-based discovery should only focus on these three features. Notably, feat_13 appeared less important in correlation analysis but emerged as the third most important feature in the random forest analysis, demonstrating the value of using multiple feature selection methods.

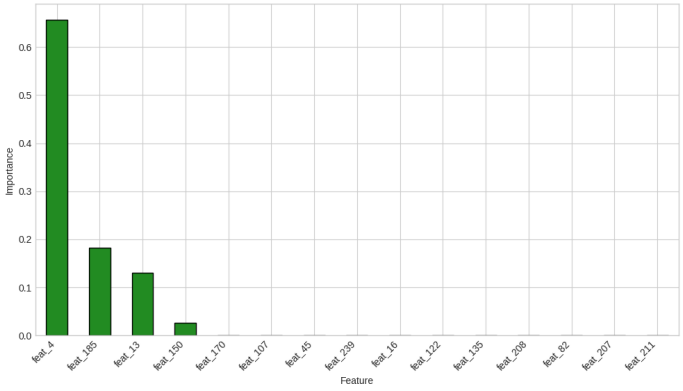

Figure 5. Few Features by Random Forest Importance

After applying two different feature identification methods, it is good to consolidate the findings and make a clear decision on which features to use for rule discovery. The best features from both methods were compared, and their combined

importance was calculated. Both features consistently identified the same dominant features: feat_4, feat_185, feat_13 together by explaining 96.94% of target02 variance. This strong verification between two independent methods provides high confidence in this feature selection. Consequently, the rule discovery process will focus exclusively on these three features.

C. Threshold Discovery

1) Decision Tree Depth Analysis:

Task 2 describes rules in an if-else structure:" if condition holds 1, then calculation1, else if condition2". This naturally maps to a decision tree, which splits the data based on threshold conditions [8]. But before building the tree, here first examined the statistics of the selected three key features to understand their value ranges. The selected three features were tested in a decision tree with varying depths (2,3,4,5) to find the optimal complexity level that balances accuracy with simplicity. Also, validation was performed to verify that the selected depth does not overfit. With the help of cross-validation confirmed no overfitting at any depth (Train, CV gap $< 1\%$). Depth 3 was selected as it provides enough accuracy (CV $R^2 = 84.7\%$) while maintaining interpretability. Since the tasks required are simple requires a shallower tree better aligns to discover simple conditions.

After finding out that depth 3 provides a good balance between accuracy and simplicity, it is important to figure out what actual threshold conditions the model discovered. This reveals the underlying logic of how target02 is determined. This tree structure showed critical insights about rule logic.The primary split on feat_4, the root node splits feat_4 $<= 0.2$, confirming that feat_4 acts as the main controller variable that determines which calculation path the follow up. With that, three distinct regions can be identified.

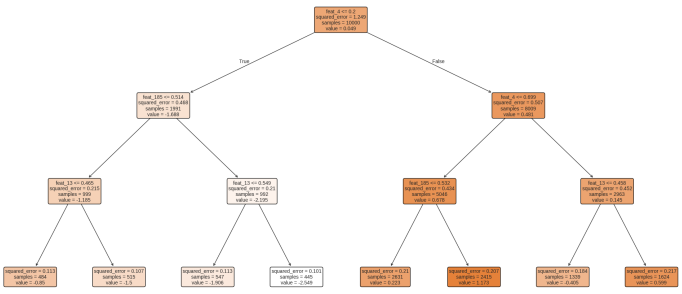

Figure 6. Decision Tree for target02 (Depth=3)

2) Region Selection:

Region 1 (feat_4 $<=0.2$) : produces negative values (-0.85 to -2.55), with feat_185 and feat_13 determining the exact value, region 2 ($0.2 <$ feat_4 $<= 0.7$) : produces mostly positive values (0.22 to 1.17), primarily driven by feat_185 region 3 (feat_4 > 0.7) produces mixed values (-0.41 to 0.6) with feat_13 becoming the deciding factor. Also, here feat_185 dominates the calculations in regions 1 and 2, while feat_13 becomes more important in region 3. This suggests the formula changes depending on the value of feat_4. These threshold values (0.2 and 0.7) and the structure switch pattern form the foundation for the final rule structure.

Figure 7 scatter plot shows a clear step-wise segmentation pattern that strongly supports or decision findings. In region 1 (feat_4$<= 0.2$): target values are predominantly negative, ranging from approximately -3 to 0. Region 2 ($0.2 <$ feat_4 $<= 0.7$): Target values shift upward, mostly positive, ranging from 0 to +2. Region 3 (feat_4 > 0.7): target values show a different pattern, spreading between -1 and +1. The graph below shows that the thresholds discovered by the decision tree (0.2 and 0.7) correspond to real structural breaks in the data.

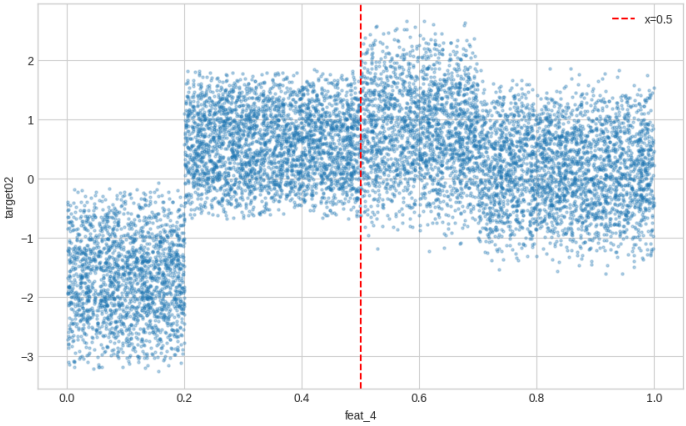

Figure 7. Feature 04 vs target02

3) Threshold Validation:

While the Decision Tree suggested thresholds at 0.2 and 0.7, it is important to systematically verify these values and understand how different thresholds affect the data segmentation. Here multiple threshold values (0.3, 0.4, 0.5, 0.6, 0.7) were tested by splitting the data into "low" and "high" regions based on feat_4. For each split, the number of samples and mean target value in each region were calculated.

Table IV
THRESHOLD VALIDATION RESULTS

Threshold	Low Region Mean	High Region Mean	Separation
0.3	-0.918	+0.465	Strong
0.4	-0.542	+0.445	Moderate
0.5	-0.317	+0.417	Moderate
0.6	-0.132	+0.317	Weak
0.7	+0.009	+0.144	Very Weak

The strongest separation occurs at lower thresholds (0.3), where the mean difference between regions is largest (approx. 1.38). As the threshold increases, the separation weakens. Combined with the Decision Tree findings (splits at 0.2 and 0.7), this confirms that multiple thresholds are needed a single

split cannot capture the full structure. The optimal approach is a 3-region model with thresholds at 0.2 and 0.7, as identified by the Decision Tree.

D. Formula Derivation

Initially, Linear Regression was applied using a simple 2 region model with a threshold at feat_4 = 0.5 (the median value). The low R^2 (for feat_4 <= 0.5, 0.597 and for feat_4 > 0.5, 0.212). This indicates that the 0.5 threshold does not align with the true data structure. This motivated the use of Decision Tree analysis to discover the correct thresholds (0.2 and 0.7), which were then used for subsequent Linear Regression analysis.

1) Linear Regression Analysis:

After identifying the thresholds (0.2 and 0.7) from Decision Tree analysis, the coefficients for each region needed to be determined. Linear Regression was used to mathematically derive initial coefficient estimates.Separate Linear Regression models were fitted for each of the three regions defined by feat_4 thresholds. The coefficients and intercepts were examined to understand the mathematical relationship.Let α and β denote the coefficients for feat_185 and feat_13 respectively:

$$(\alpha, \beta) = \begin{cases} (-2, -1) & \text{if } feat\_4 \leq 0.2 \\ (+2, -1) & \text{if } 0.2 < feat\_4 \leq 0.7 \\ (-1, +2) & \text{if } feat\_4 > 0.7 \end{cases} \quad (1)$$

This piecewise model achieves an overall $R^2 = 0.9067$.

2) Grid Search Analysis:

Direct rounding of Linear Regression coefficients may not yield optimal results due to noise in the data. Therefore, Grid Search was employed to verify and refine the integer coefficients [9].The search space included all integer coefficients in $\{-2, -1, 0, 1, 2\}$ for both feat_185 and feat_13 across all regions. Two model configurations were evaluated:

- 2-region model: $R^2 = 0.375$
- 3-region model: $R^2 = 0.915$

The significant improvement confirms that the 3-region piecewise structure better captures the underlying relationship.

But linear regression in Region 3 found coefficients (-0.76, +1.97), achieving R² = 0.8358 with decimal values. However, when rounded to integers (-1, +2), performance dropped significantly to R² = 0.5276. Grid Search revealed that (-1, +1) performs better (R² = 0.6052), suggesting that the +1.97 coefficient was influenced by noise. This demonstrates the importance of validating rounded coefficients rather than blindly rounding Linear regression results.

E. Final Validated Rules

Linear Regression provided initial coefficient estimates for each region. Grid Search was then employed to validate and optimize these estimates, ensuring robustness against noise-influenced values. The final piecewise model achieving $R^2 = 91.50\%$ can be expressed as where α and β denote the coefficients for feat_185 and feat_13 respectively.

$$(\alpha, \beta) = \begin{cases} (-2, -1) & \text{if } feat\_4 \leq 0.2 \\ (+2, -1) & \text{if } 0.2 < feat\_4 \leq 0.7 \\ (-1, +1) & \text{if } feat\_4 > 0.7 \end{cases} \quad (2)$$

where α and β denote the coefficients for feat_185 and feat_13 respectively, achieving an overall $R^2 = 91.50\%$. Additionally, a grid search including feat_4 as a calculation variable (rather than a threshold controller) achieved only $R^2 = 0.432$. This confirms that feat_4 functions purely as a region selector, not a direct predictor, validating the 3-region piecewise approach.

F. Model Validation

The Actual vs Predicted plot confirms strong predictive accuracy ($R^2 = 0.915$), while the Residual plot shows randomly distributed errors centered at zero, validating the model's assumptions.

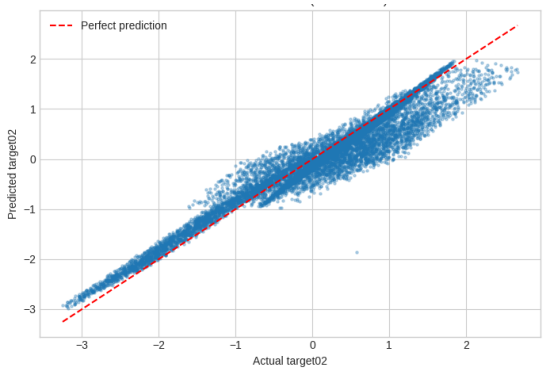

Figure 8. Actual vs Predicted target02

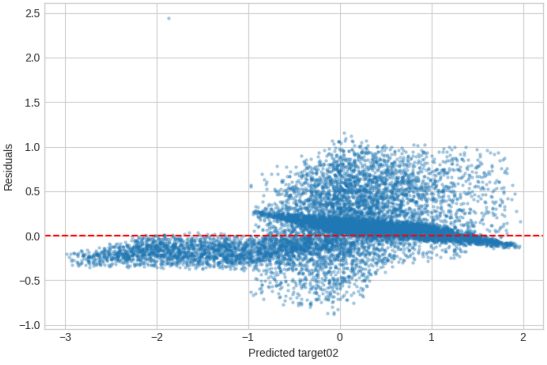

Figure 9. Residual Plot

G. Evaluation Dataset Validation

The discovered rules were applied to the evaluation dataset (`EVAL_81.csv`) containing 10,000 samples.Table V shows the distribution of samples across regions.

Table V
EVALUATION DATASET DISTRIBUTION BY REGION

Region	Condition	Samples	Percentage
1	$f_4 \leq 0.2$	2,005	20.1%
2	$0.2 < f_4 \leq 0.7$	4,930	49.3%
3	$f_4 > 0.7$	3,065	30.6%
	Total	**10,000**	**100%**

The distribution closely mirrors the training data (20%, 50%, 30%), confirming that the evaluation data follows the same underlying structure and supporting the generalizability of the discovered rules.

H. Edge Device Implementation

The discovered rules were implemented in framework_81.py using the customer's specified condition-calculation pair structure. The implementation uses only basic Python operations and permitted libraries (`numpy`, `pandas`, `operator`, `argparse`), satisfying all edge device deployment constraints. Conditions are evaluated sequentially, with the first matching condition's calculation applied:

Table VI
FRAMEWORK IMPLEMENTATION RULES

Priority	Condition	Calculation
1	$feat\_4 \leq 0.2$	$-2 \times feat\_185 - feat\_13$
2	$0.2 < feat\_4 \leq 0.7$	$+2 \times feat\_185 - feat\_13$
3	$feat\_4 > 0.7$	$-feat\_185 + feat\_13$

IV. CONCLUSION

Two complementary approaches were developed for predictive modeling, tailoring the solution to the specific nature of each target variable. For Task 1, where the data exhibited high noise and complex nonlinear dependencies, a Stacking Ensemble was required. By combining gradient boosting models with a Ridge meta learner, this approach achieved $R^2 = 0.844$, demonstrating a significant improvement over baseline methods. For Task 2, exploratory analysis revealed a distinct logical structure. By following a systematic rule discovery process, feature identification, threshold discovery, and segmented regression successfully reverse engineered the underlying formula. This yielded a highly accurate ($R^2 = 0.915$) and fully interpretable model.

V. ACKNOWLEDGMENTS

This work was implemented using Python with scikit-learn and gradient boosting libraries (XGBoost, LightGBM, CatBoost) for model development, and Optuna for hyperparameter optimization. Claude (Anthropic) was used for code debugging assistance and editorial refinement. All analysis and conclusions remain the author's original work.

REFERENCES

[1] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, Aug 2005.

[2] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. San Francisco, CA, USA: ACM, 2016.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *arXiv preprint arXiv:1907.10902*, Jul 2019. [Online]. Available: https://arxiv.org/abs/1907.10902

[4] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Long Beach, CA, USA: Curran Associates, Inc., 2017.

[5] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*. Montreal, Canada: Curran Associates, Inc., Dec 2018.

[6] M. N. Islam, M. M. H. Rimon, S. S.-E.-A. Shamim, Z. M. Fahad, M. J. I. Mony, and M. J. U. Chowdhury, "An improved ensemble-based machine learning model with feature optimization for early diabetes prediction," nov 2025. [Online]. Available: https://arxiv.org/abs/2512.02023

[7] GeeksforGeeks, "Feature importance with random forests," 2024, accessed: 2025-01-27. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/feature-importance-with-random-forests/

[8] C. Molnar, "Interpretable machine learning: A guide for making black box models explainable," 2025, chapter 6.2: Decision Rules. Accessed: 2025-01-27. [Online]. Available: https://christophm.github.io/interpretable-ml-book/rules.html

[9] J. Brownlee, "Hyperparameter optimization with random search and grid search," 2020, machine Learning Mastery. Accessed: 2025-01-27. [Online]. Available: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/