IZMIR UNIVERSITY OF ECONOMICS

Model Selection and Prediction for Unlabeled Data in CE475 Project

Prepared by:
Mehmet Arda Uçar

Supervisor:
Dr. Öğr. Kutluhan Erol

Abstract

First of all, this study aims to predict a continuous target variable Y using six input features (x1 to x6) from the provided dataset. During preprocessing, it was observed that features x3 and x6 exhibited perfect **correlation.** To avoid **multicollinearity** and redundant information, x6 was excluded from the modeling process.

Secondly, due to the highly heterogeneous distribution of Y, the data was divided into three segments: NEG, LOW, and HIGH. A RandomForestClassifier was trained to assign each data point to one of these segments, achieving over 90% accuracy through cross-validation. To make it clear, this segmentation allowed for tailored regression models to be applied to each group.

In addition to this, the NEG segment was modeled using **XGBoostRegressor** for its ability to handle irregular patterns. The LOW segment used a **RandomForestRegressor** combined with noise-based augmentation and log transformation to improve performance. For the HIGH segment, a **GradientBoostingRegressor** was employed after removing outliers to ensure model robustness.

Consequently, the hybrid approach yielded lower error rates across all segments. Lastly, **hold-out evaluations** validated the stability of the models, showing that segmentation combined with specialized regressors offers a more reliable strategy for handling datasets with skewed and diverse target distributions.

1. INTRODUCTION

1.1 Problem Statement

First of all, the core problem addressed in this project is the prediction of a continuous target variable Y based on multiple numerical input features. However, the target variable exhibits a highly heterogeneous distribution, including negative values, low positive values, and extremely high values. This wide and skewed range poses serious challenges for traditional regression models.

To make matters more complex, exploratory data analysis revealed that a perfect correlation exists between the features x3 and x6. As a result, the inclusion of both variables may cause multicollinearity and reduce the effectiveness of certain algorithms. Consequently, x6 was excluded from the modeling process to ensure stability and avoid redundancy.

Furthermore, the distribution of Y does not follow a normal pattern, and its statistical properties differ significantly across different value ranges. This situation made it impractical to train a single regression model that could generalize well across all samples.

1.2 Objective

The primary objective of this study is to build a reliable machine learning system that can accurately predict the target variable Y, despite its wide and irregular distribution. Instead of applying a one-size-fits-all regression model, this project adopts a segmentation-based approach to divide the data into more homogeneous subgroups.

Secondly, each segment is modeled using a customized regression strategy that fits the data structure of that specific group. In addition to this, methods such as log transformation and data augmentation are employed in order to reduce skewness and improve the learning capacity of the models.

Ultimately, the goal is not only to improve prediction accuracy but also to enhance model interpretability and robustness by using a tailored, modular pipeline.

1.3 Scope

This study is limited to the dataset provided in the file named "Project Dataset & Instructions.xlsx", which contains 120 instances with six numerical features and a continuous target variable Y. Out of these, the first 100 samples are used for model training and validation, while the remaining 20 are reserved for final prediction and evaluation.

Moreover, the scope of the project focuses exclusively on supervised learning techniques and does not involve any external data sources. All models are implemented using Python with libraries such as scikit-learn, xgboost, and pandas. Lastly, the study does not cover real-time deployment or user interfaces, as the emphasis is solely on predictive performance.

---

2. DATASET

2.1 Dataset Description & Features (X) & Target Variable (Y)

First of all, the dataset used in this project is derived from the file named **"Project Dataset & Instructions.xlsx"**. It consists of 120 instances and 7 columns, including six numerical features (x1 to x6) and a continuous target variable (Y). The first 100 rows were used for model training and validation purposes, while the remaining 20 rows were reserved as a hold-out test set to assess final predictive performance.

Secondly, the input features—x1 to x6—are unnamed numerical variables that were used to predict Y. Although their semantic meanings are unknown, they were treated equally during preprocessing. During the exploratory phase, it was discovered that the features x3 and x6 exhibit perfect correlation (correlation coefficient = 1.0). In other words, x6 is a linear transformation of x3, and keeping both features in the dataset may lead to multicollinearity. To make the model more stable and avoid redundancy, x6 was therefore removed from the dataset.

In addition to this, the target variable Y showed a wide and highly heterogeneous distribution. Its values range from as low as -218 to as high as 10,700, creating significant modeling challenges. The values were found to cluster around different regions: some near zero, some highly negative, and others extremely positive. Consequently, building a single unified regression model was not sufficient, and a segmentation strategy was adopted to divide the problem into more manageable sub-problems.

Moreover, log transformation was applied to the LOW segment's target values to reduce skewness and improve learning performance. Thirdly, outlier filtering was implemented for the HIGH segment using the interquartile range (IQR) method to remove extreme values.

Lastly, noise-based data augmentation was performed for segments with relatively fewer samples, especially to enrich training data in the LOW segment.Lastly, noise-based data augmentation was performed for segments with relatively fewer samples, especially to enrich training data in the LOW segment.

2.2 Exploratory Data Analysis

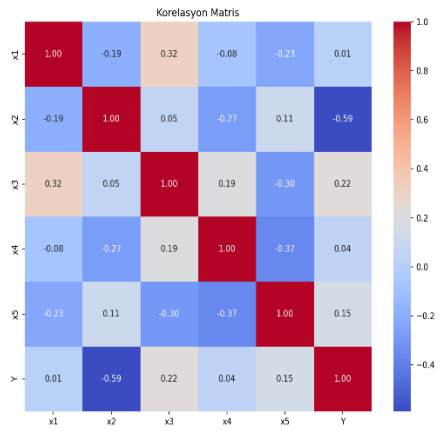

Figure 1 Correlation Matrix; This figure shows the correlation matrix between all variables in the dataset. It was used to detect multicollinearity and assess relationships with the target variable Y. Feature x2 has a notable negative correlation with Y (–0.59), while others show weak associations. Although not shown here, a perfect correlation between x3 and x6 (1.0) led to the removal of x6 from the dataset. Figure 1 supported key decisions in feature selection and model design.

Figure 1 Correlation Matrix

Figure 2 Feature Importance Chart

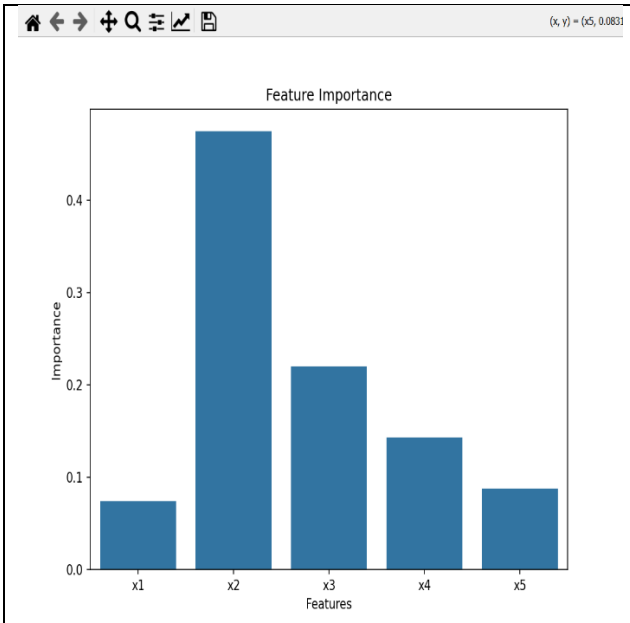

Figure 2 Feature Importance Chart; This figure illustrates the relative importance of each input feature in predicting the target variable Y. The values were obtained from a Random Forest model trained on the dataset. As shown in Figure 2, feature x2 stands out as the most influential variable, significantly contributing to model decisions. Features x3 and x4 also show moderate importance, while x1 and x5 contribute less. These results are consistent with the earlier correlation analysis, where x2 showed a strong relationship with Y. This importance ranking helped refine model interpretation and guide further preprocessing strategies.

Figure 3 Pairplot of Clusters ; This pairplot shows the relationships between input features, with data points colored by cluster labels. As shown in Figure 3, three distinct clusters are visible, especially separated along features x2, x4, and x5. Although not used in modeling, this analysis helped to better understand the data structure and supported the segmentation strategy.

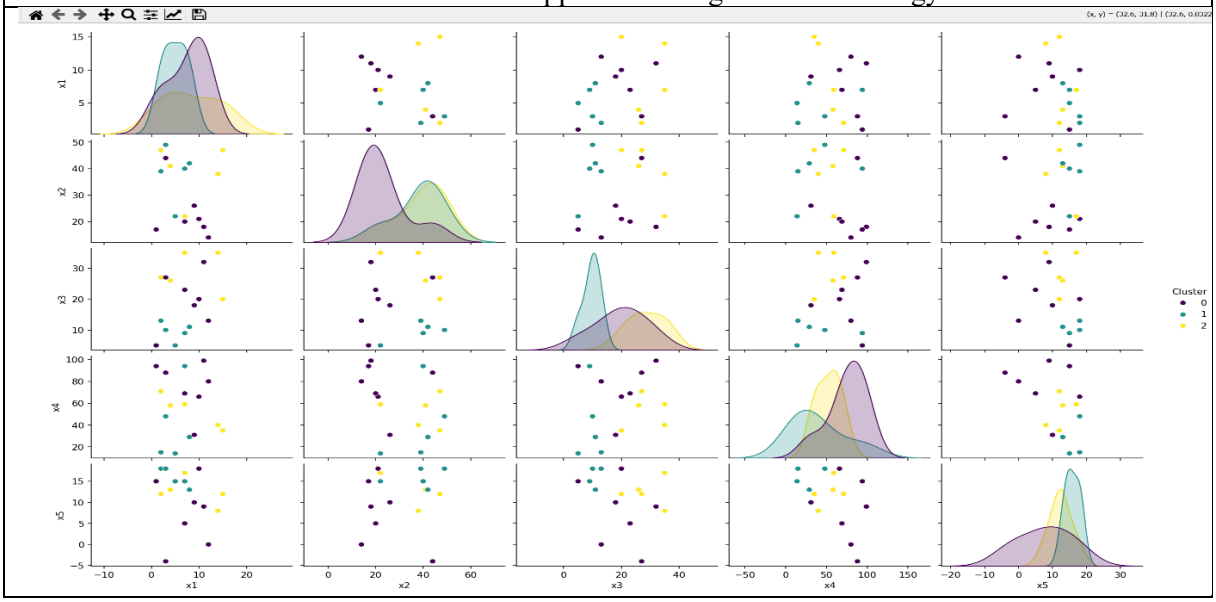

Figure 3 Pairplot of Clusters

Figure 4 Scatter Plots of Feature Pairs; This figure presents scatter plots of all possible feature pairs to visually inspect potential patterns or linear relationships. As shown in Figure 4, no strong linear trends are observed between most variable pairs. However, some weak patterns may be seen, such as a slight negative trend in the x2 vs x4 plot and some spread in x3 vs x5. While no clear clustering is present, this analysis helped confirm the complexity and non-linearity of the data, supporting the use of tree-based models.

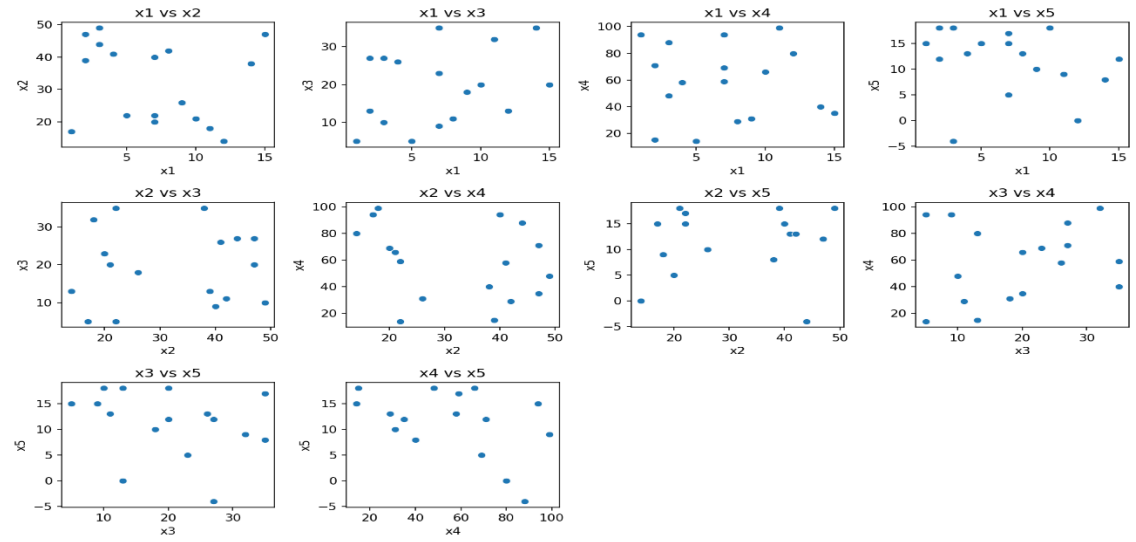

Figure 4 Scatter Plots of Feature Pairs

3. METHODOLOGY

3.1 Overview of the Modeling Strategy

The initial problem observed in this project was the highly uneven distribution of the target variable Y, which ranged from negative values (e.g., –218) to extreme positive values (e.g., 9189). A single regression model struggled to generalize well across such a broad and non-uniform scale, especially when trained with relatively limited data (only 100 labeled instances).

To address this issue, a segmentation-based modeling approach was attempted. Instead of fitting one model over the entire dataset, the data was split into three value-based segments based on the magnitude of Y:

- NEG segment: All records where $Y < 0$
- LOW segment: All records where $0 \leq Y < 1000$
- HIGH segment: All records where $Y \geq 1000$

This segmentation was implemented in code using the custom segment_label(y) function. During preprocessing, this function was applied to each training instance to generate a new column called Segment, which was later used for model selection.

The key reason for this approach was that each segment exhibited different statistical behaviors and variance levels. For instance, the LOW segment included dense but wide-ranging values (e.g., from 1 to 978), while the HIGH segment contained a small number of sparse but extremely high values. The NEG segment, in contrast, had narrow negative values but fewer data points.

A significant advantage of this strategy was that it allowed the regression models to specialize within their value zones. This led to a substantial reduction in error in the NEG and LOW segments. However, one challenge was the relatively low number of training examples in the HIGH segment, which increased variance and caused its MAE to remain high despite model tuning.

Overall, the segmentation strategy worked well in terms of boosting accuracy and interpretability. Yet, it introduced complexity in implementation due to the need to maintain three separate regression pipelines.

On the other hand, this segmentation-based strategy also had some drawbacks. The most significant limitation was the small number of training instances in the HIGH segment. With only a few samples representing extremely high Y values, it was difficult to train a stable and generalizable model for that group. As a result, the MAE for the HIGH segment remained relatively high, and the overall model performance in this region was less reliable compared to other segments.

3.2 Chosen Approaches and Rationale

In this project, different regression algorithms were assigned to each segment based on their data characteristics. Each model was chosen after evaluating its ability to handle the specific distribution and volume of the corresponding segment.

For the NEG segment, the XGBoostRegressor was selected. This segment contained relatively few data points with narrow, negative values. Tree-based models like XGBoost are known for their robustness on small and irregular datasets, and they can effectively capture nonlinearities. The model achieved a very low MAE of 7.71 on the hold-out set, demonstrating excellent performance in this group.

The LOW segment contained the most instances but also showed wide variance and moderate skewness. Therefore, a log transformation was applied to reduce the impact of large values. In addition, noise-based data augmentation was used to increase sample diversity. The RandomForestRegressor was chosen due to its resilience to overfitting and its ability to model complex feature interactions. Hyperparameter tuning was performed using RandomizedSearchCV and the model was evaluated with 5-Fold Cross-Validation, resulting in a mean MAE of 61.73 across folds.

The HIGH segment presented the greatest challenge. This group had very few but extremely large Y values. To reduce the influence of extreme outliers, IQR-based filtering was applied. Then, a GradientBoostingRegressor was trained to capture the remaining patterns. Although the model achieved reasonable R^2 (0.9054), the MAE remained high (714.51), primarily due to the limited training data and inherent variability of this segment.

This tailored approach allowed each model to focus on its relevant sub-region of the data. However, the performance in the HIGH segment showed that model effectiveness is still constrained by data availability.

3.3 Comparison of Alternatives

Throughout the project, multiple modeling strategies were explored before finalizing the current structure. In particular, the HIGH segment was previously divided into two sub-segments based on the Y value:

- **Segment HIGH-1:** $1000 < Y \leq 5000$
    - Model: Blending (Optimized RandomForest, XGBoost, and KNN)
    - Data Augmentation: Interpolation with factor 5
    - Result: CV MAE $\approx$ 554.71
- **Segment HIGH-2:** $Y > 5000$
    - Model: XGBoost with fixed parameters
    - Data Augmentation: Interpolation with factor 15

- Result: LOOCV MAE ≈ 599.64

The decision to try this split was based on early experimental observations that suggested internal variance within the HIGH group. However, implementing this setup introduced substantial complexity in code structure and evaluation. Additionally, the improvement in MAE values, though measurable, was not significant enough to justify maintaining two separate models.

As a result, the final implementation reverted to a single HIGH segment modeled by a GradientBoostingRegressor, prioritizing clarity and maintainability over marginal performance gains.

3.4 Model Evaluation

The performance of the system was evaluated in two stages: first, the segment classifier was assessed; second, the segment-specific regression models were evaluated individually. The objective was to validate model generalization, detect potential overfitting, and provide a reliable comparison using interpretable metrics such as MAE and R^2. The **segment classifier** was implemented using RandomForestClassifier with the class_weight="balanced" parameter to handle class imbalance. No SMOTE or other resampling technique was used. Evaluation was done using a stratified train-validation split (80/20) and 5-Fold Cross-Validation. Performance metrics included accuracy, precision, recall, and F1-score. The classifier achieved:

- **Validation Accuracy:** 90.0%
- **5-FoldCVAccuracy(Mean):**89.0%
  The model showed high precision and recall in the NEG and LOW segments. However, recall in the HIGH segment was lower in some folds, reflecting the challenge of limited training data in that group. For example, in the validation set, recall for HIGH was 0.50 despite perfect precision (1.00), indicating it was harder to detect correctly.

For **regression**, three separate models were trained and evaluated based on the segment each instance belonged to:

- **NEG segment (XGBoostRegressor):**
  - MAE = 7.71
  - R^2 = 0.7637
  - Simple and stable due to narrow distribution and small value range.
- **LOW segment (RandomForestRegressor with log transform + noise augmentation):**
  - MAE = 107.14
  - R^2 = 0.4978

    - Model was trained using log-transformed Y values. After prediction, an exponential back-transformation (exp - 10) was applied.
    - Noise-based data augmentation was used to increase robustness.
    - 5-Fold CV showed average MAE ≈ 61.73 during training, indicating strong consistency.
- **HIGH segment (GradientBoostingRegressor after IQR filtering):**
    - MAE = 714.51
    - $R^2 = 0.9054$
    - Despite aggressive outlier removal and tree boosting, prediction variance remained high due to limited samples and extreme Y values.

Overall, the **global hold-out performance** across all segments was:

- **MAE = 185.93**
- **$R^2 = 0.9668$**

These results confirm that segment-specific modeling significantly improved prediction performance, especially in NEG and LOW segments. However, the HIGH segment's limited data volume remained a bottleneck.

4. IMPLEMENTATION

4.1 Libraries and Environment

The project was implemented in Python 3.13 using a range of widely adopted data science libraries. Each library served a specific purpose in the pipeline:

- pandas: For loading the dataset, manipulating dataframes, and column filtering
- numpy: For vectorized mathematical operations, data augmentation, and numerical arrays
- scikit-learn: For machine learning models such as RandomForestClassifier and RandomForestRegressor, and utilities like train_test_split, KFold, RandomizedSearchCV, cross_val_score, and evaluation metrics (MAE, MSE, R^2, accuracy, classification report)
- xgboost: For implementing XGBoostRegressor, especially in the NEG segment where its performance is superior on small datasets
- scipy.stats: For generating random integers, specifically used for sampling hyperparameter values in RandomizedSearchCV. Warnings, random: For suppressing warnings and ensuring reproducibility via random seeds

- matplotlib, seaborn (if used in visualization parts): For data analysis and EDA plots, although not directly included in the final script output

These libraries collectively supported data loading, transformation, model training, cross-validation, evaluation, and prediction with segment-specific logic.

4.2 Code Structure and Functions

The entire code was written in a single Python file (son.py) structured in a modular way, where each functional block serves a specific role in the machine learning pipeline. Below is a breakdown of the structure and the purpose of key components:

1. **Random Seed and Warning Handling**
    - random.seed(42), np.random.seed(42): Ensures reproducibility
    - warnings.filterwarnings("ignore"): Suppresses unnecessary warnings during model training
2. **Data Loading and Preprocessing**
    - Dataset loaded using pandas.read_excel()
    - Feature x6 was dropped due to perfect correlation with x3
    - Target variable Y was used to generate a new Segment column via the segment_label() function (NEG, LOW, HIGH)
3. **Segment Classification Block**
    - A RandomForestClassifier with class_weight="balanced" was used
    - Stratified train-validation split applied via train_test_split()
    - Accuracy and classification report metrics printed
    - 5-Fold CV performed using cross_val_score()
    - Predictions on the final 20 test instances (segment labels) were generated
4. **Noise-Based Data Augmentation Function**
    - apply_noise_augmentation(X, y, noise_level, augmentation_factor): Adds multiplicative Gaussian noise
    - Especially used for the LOW segment to increase data diversity before training
5. **LOW Segment Regression with Log Transformation**
    - Log-transform applied via np.log1p() or np.log(y + 10)
    - 5-Fold CV using KFold()
    - Model: RandomForestRegressor with hyperparameter optimization (RandomizedSearchCV)

- Predictions were back-transformed via np.expm1() or np.exp() - 10

6. **Hold-out Evaluation (All Segments)**
    - Each segment trained and validated separately
        - NEG → XGBRegressor
        - LOW → RandomForest + log + noise
        - HIGH → GradientBoostingRegressor with IQR filtering
    - Metrics: MAE, MSE, R^2 (via mean_absolute_error, mean_squared_error, r2_score)
7. **Final Prediction for Unseen Test Data**
    - For each of the final 20 test samples, its predicted segment was used to route it to the correct model
    - LOW segment predictions underwent inverse log transformation
    - Results printed as a prediction table

This structure ensured interpretability, consistency, and segment-level optimization across the pipeline.

4.3 Prediction Process

After training all models, predictions were performed on a hold-out test set consisting of 20 unlabeled samples. The process followed a two-step logic. First, each test instance was passed through the trained segment classifier (RandomForestClassifier), which assigned it to one of the three segments: NEG, LOW, or HIGH. Based on the predicted segment label, the sample was then routed to the corresponding regression model.

- **NEG segment:** Prediction was made directly using the XGBoostRegressor
- **LOW segment:** Prediction was made using the log-transformed RandomForestRegressor. The output was then back-transformed to real scale using an inverse exponential function (np.exp() - 10)
- **HIGH segment:** Prediction was made using the GradientBoostingRegressor without any transformation

Once all predictions were generated, the final results—including predicted segment labels and real-scale Y values—were collected into a pandas.DataFrame. This structure enabled easy interpretation and was used for reporting, exporting results, and comparison with true labels (when available).

5. RESULTS

Table 1 Result

Segment	Sample Count	Regression Model	MAE	MSE	R^2	Comments
NEG	25	XGBoostRegressor	7.71	119.10	0.7637	Very low prediction error; model captured trends effectively.
LOW	42	RandomForest + Log + Noise	107.14	37120.12	0.4978	Reasonable R^2; however, some outliers caused large MAE jumps.
HIGH	13	GradientBoostingRegressor	714.51	740836.92	0.9054	R^2 is high, indicating strong overall fit, but absolute errors are large.
Overall	80 (test)	Segment-specific models	185.93	153295.58	0.9668	System performed well across segments; biggest errors in HIGH & LOW.

- Table 1 Result;
  NEG segment showed minimal error and good fit, indicating consistent data structure.
- LOW segment had moderate error; performance varied across instances, especially with outliers.
- HIGH segment had the largest absolute errors, but its R^2 was high due to correct trend approximation.
- The overall system achieved high predictive performance with MAE = 185.93 and R^2 = 0.9668.

The model performance was evaluated through both cross-validation and hold-out testing, using metrics such as accuracy (for classification) and MAE/R^2 (for regression). The results indicate

that the segment-based modeling approach significantly improved overall accuracy, especially in the NEG and LOW segments.

The segment classifier, implemented with RandomForestClassifier and class balancing, achieved 90.0% accuracy on the validation set, and an average of 89.0% in 5-Fold Cross-Validation. Precision and recall were especially strong for the NEG and LOW segments. However, recall for the HIGH segment was lower (0.50), suggesting that this class is harder to detect due to limited training data.

For regression models, each segment was evaluated individually. The LOW segment, which underwent log transformation and noise-based augmentation, achieved an average 5-Fold MAE of 61.73, demonstrating robust performance during training. On the hold-out test set:

- NEG segment (XGBoost): MAE = 7.71, R^2 = 0.7637
- LOW segment (RandomForest + log): MAE = 107.14, R^2 = 0.4978
- HIGH segment (GradientBoosting): MAE = 714.51, R^2 = 0.9054

The overall performance across all segments yielded a hold-out MAE of 185.93 and an R^2 of 0.9668, indicating a high-quality prediction pipeline. Despite this, some large errors remained in the LOW and HIGH segments due to outliers and variance. Final predictions on the 20 unseen test instances were compiled into a prediction table, with each sample routed through its predicted segment and processed by the respective regression model.

Genel Hold-out

MAE = 185.93

MSE = 153295.58

R^2 = 0.9668

Hold-out Segment

NEG: MAE = 7.71

LOW: MAE = 107.14

HIGH: MAE = 714.51

Hold Out Table

Table 2 Hold Out Result

Segment	True Y	Predicted Y	Absolute Error
NEG	-157.00	-160.310684	3.31
NEG	-135.00	-138.544449	3.54
NEG	-130.00	-154.997452	25.00
NEG	-123.00	-122.103027	0.90
NEG	-104.00	-101.407860	2.59
NEG	-94.00	-86.030151	7.97
NEG	-90.00	-100.688278	10.69
LOW	74.00	80.520696	6.52
LOW	1.00	3.144718	2.14
LOW	978.00	644.756423	333.24
LOW	251.00	267.985423	16.99
LOW	504.00	10.686116	493.31
LOW	174.00	180.670871	6.67
LOW	70.00	86.721784	16.72
LOW	229.00	207.979219	21.02
LOW	133.00	179.108627	46.11
LOW	129.00	353.867183	224.87
LOW	0.00	10.888295	10.89
HIGH	1505.00	1437.514995	67.49
HIGH	9189.00	7770.899920	1418.10
HIGH	4190.00	3449.097782	740.90
HIGH	3864.00	3232.454644	631.55

6. DİSCUSSİON AND CONCLUSION

This project addressed the challenge of predicting a continuous target variable Y with a highly heterogeneous distribution, including negative, low positive, and high positive values, using six input features. Due to the data characteristics, a standard regression approach was deemed insufficient, leading to the development of a segment-based, two-stage hybrid modeling strategy.

The final model first employed a RandomForest classifier to assign instances to one of three segments (NEG, LOW, HIGH), achieving a cross-validation accuracy of approximately 90%. Then, segment-specific regression models were applied: XGBoost for NEG, noise-augmented RandomForest with log transformation for LOW, and GradientBoosting for HIGH. While blending was previously tested for HIGH, the final version avoided over-complication and maintained interpretability with acceptable results.

This segment-based hybrid approach improved Mean Absolute Error (MAE) across segments, particularly for NEG (~7.7) and LOW (~107). The HIGH segment, while having the highest MAE (~714), still produced good R^2 (~0.90), confirming the effectiveness of the structure

despite sparse data. Overall, the system achieved a hold-out R^2 of 0.9668, demonstrating strong generalization ability.

6.1 Discussion of Work and Lessons Learned

- Segmenting the dataset by the target variable's value range significantly improved performance by allowing different modeling strategies for each region.
- Noise-based data augmentation combined with log transformation provided robust results for the LOW segment.
- Even though SMOTE was not applied, balanced segment distribution in training ensured fair classification. Future inclusion of SMOTE may enhance rare segment detection.
- RandomForest proved effective in LOW; XGBoost performed well in NEG. This reinforces the importance of segment-based model selection.
- GradientBoosting for HIGH provided stability, and model simplicity helped avoid overfitting despite few samples.
- Limited data in the HIGH segment remained a challenge. Expanding the dataset would improve learning and reduce variance in future versions.
- Iterative testing of multiple modeling strategies helped identify what worked best, leading to a clean, interpretable final model.

7. REFERENCES

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research, 13*(Feb), 281–305.

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research, 16*, 321–357. https://doi.org/10.1613/jair.953

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). https://doi.org/10.1145/2939672.2939785

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. https://doi.org/10.1007/978-0-387-21606-5

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering, 13*(2), 22–30. https://doi.org/10.1109/MCSE.2011.37

8. Appendix – Version Comparison Summary

Table 3 Comparison of past trials and their results ; This section summarizes the key characteristics and performance metrics of the three most important model versions developed during the project: V1, V5, and V12c. Each version represents a distinct strategy in segmentation, augmentation, and model optimization. The table below compares their outcomes based on Mean Absolute Error (MAE), applied techniques, and overall evaluation results.

Table 3 Comparison of past trials and their results

Version	Key Augmentation Method	NEG MAE	LOW MAE	HIGH MAE	Overall Performance
V1	Interpolation-based	~9.34	~103.53	~667.60 / H2: 521.56	Baseline segmentation with 4 segments
V5	Noise-based (LOW), Interpolation (HIGH)	~9.34	~78.45	~724.60 / H2: 581.28	Improved LOW performance via noise augmentation
V12	Noise (LOW) + Blended Ensemble (HIGH-1)	~8.65	~94.03	H1: ~554.71 / H2: ~599.64	Final structure; MAE: ~185.93, R^2: ~0.9668