MGSC – 695
Final Project: Multi-task Learning for Predicting House Prices and House Category

Prepared for

Fatih Nayebi & Necmiye Gen

Yichen Yu

April 2024

Table of Contents

Abstract

This study documents the evolution of a multi-task learning model engineered to concurrently estimate residential property values and classify homes into distinct categories. Leveraging a comprehensive dataset, the model harnesses the shared and unique patterns across regression and classification tasks to improve generalization. Initial explorations involved preprocessing a feature-rich dataset, addressing missing values, and normalizing the data. A series of experiments with various activation functions, optimizers, and advanced PyTorch Lightning features informed iterative enhancements, with the objective to strike a balance where enhancements to one task did not impede the performance of the other.

Central to the architecture was a shared bottom layer for feature extraction, which fed into task-specific top layers for regression and classification. Advanced features like logging, model checkpointing, and early stopping were employed to streamline the training process and mitigate overfitting. Hyperparameter tuning with Optuna was a crucial step that significantly improved model performance. The final model, characterized by its optimal hyperparameters, achieved a Mean Squared Error of 0.1549 and a classification accuracy of 0.88, marking a substantial improvement over earlier versions.

Despite the model's high accuracy, it recognized challenges such as class imbalance and the complexity of feature relationships, pointing to areas for potential improvement. The research underscores the promise of multi-task learning in real estate analytics while offering a candid account of the iterative nature of model development and the quest for refinement. This multi-faceted learning journey not only improved predictive accuracy but also provided valuable insights for future enhancements in the field.

1. Introduction

1.1. Purpose of Multi-Task Learning

Multi-task learning (MTL) is a subfield of machine learning where multiple learning tasks are solved simultaneously, leveraging commonalities and differences across tasks to improve generalization. In the context of real estate, an MTL approach can simultaneously predict house prices and categorize houses, which can be beneficial as these tasks can share underlying patterns related to house features, market trends, and buyer preferences. By sharing representations between related tasks, an MTL model can gain a more robust understanding, potentially leading to more accurate and generalizable predictions than separate models trained on individual tasks.

1.2. Overview of the Dataset and Prediction Tasks

The dataset utilized in this project is the "House Prices - Advanced Regression Techniques" dataset, which comprises various attributes of houses, including physical characteristics, location details, and historical sales data. From this rich dataset, a new categorical variable 'House Category' is engineered by combining attributes such as 'House Style', 'Bldg Type', 'Year Built', and 'Year Remod/Add'. The regression task aims to predict house prices, a continuous variable directly indicative of market value. In parallel, the classification task seeks to assign houses to distinct categories based on their characteristics and renovation history, providing a qualitative measure of the property's market segment.

1.3. Project Goals and Expected Outcomes

The goal of this project is to construct and fine-tune a multi-task learning model that can efficiently predict house prices while also categorizing properties into meaningful groups. This dual objective serves to broaden the applicability of the model, making it a valuable tool for various stakeholders, including home sellers, buyers, and real estate analysts. Expected outcomes include achieving a balance between the two tasks where improvements in one do not detrimentally affect the performance of the other. By effectively harnessing the capabilities of PyTorch Lightning, the project aims to streamline complex model management, allowing for seamless experimentation and iterative improvement. Success will be measured by the model’s regression metrics, accuracy, and F1 score in the respective tasks, along with the insights generated from the data through this multi-faceted lens.

2. Data Exploration and Preprocessing

2.1 Dataset Overview

The dataset comprises 1,460 properties, each described by 80 features that include a mix of numerical and categorical data, indicating varied house attributes. The initial exploration indicates that certain features, such as 'LotFrontage', have missing entries, necessitating careful preprocessing.

2.1.1. The Distribution of Sale Price

The 'SalePrice' variable in the dataset exhibits a right-skewed distribution in Figure 1, signaling a concentration of homes within the more affordable price brackets and fewer instances of higher-valued luxury properties. This skewness suggests that median sale price might provide a more accurate measure of central tendency than the mean, which is likely to be influenced by the expensive outliers. To address this and enhance model performance, a logarithmic transformation of 'SalePrice' is considered, normalizing the distribution for the regression task. The observed distribution underscores the importance of selecting robust evaluation metrics for the models that are less sensitive to outliers, ensuring that the predictive accuracy reflects a true representation of the market's dynamics.

2.2 Missing Value Analysis

An analysis of missing values leads to the elimination of columns with more than half of the data absent. Remaining gaps are filled using the median for numerical columns and mode for categorical ones, ensuring a complete dataset for model input.

2.3 Feature Engineering

Throughout preprocessing, the objective remains consistent: to clean and transform the data into a suitable format for developing a robust multi-task learning model. This involves crafting a 'HouseCategory' target for classification and preserving 'SalePrice' for regression, ensuring both tasks can be trained simultaneously yet effectively.

The 'HouseCategory' variable is an engineered feature that classifies houses into distinct categories based on their architectural style, type, age, and remodeling history. This categorization is accomplished through a function that combines four key property attributes: 'House Style', 'Building Type', 'Year Built', and 'Year Remod/Add'. Each attribute contributes to a subcategory—style, type, age, and remodel status—which are then concatenated to form a comprehensive 'HouseCategory' descriptor for each property. For example, a multi-story, single-family home built after 2000 and not remodeled would be categorized as "Multi-Story, Single-Family, Modern, Not Remodeled".

2.4 Categorical Variable Encoding

Categorical features undergo one-hot encoding, transforming them into a format suitable for model training. The custom 'HouseCategory' variable is excluded from this step to preserve its newly assigned categorizations.

2.5 Data Normalization

Numerical features are standardized using StandardScaler, aligning the dataset to a common scale and preparing it for the sensitive algorithms of the multi-task learning model.

3. Multi-task Model Building

3.1 Architectural Overview of the Multi-Task Learning Model

The MultiTaskModel is crafted with the innovative approach of multi-task learning, allowing it to simultaneously predict house prices (regression) and classify house categories (classification). Its architecture is designed to efficiently handle the intricacies of dual-task prediction within a singular framework, facilitating shared learning and a compact model structure that aims to exploit commonalities between tasks.

3.2 The Shared Bottom Model: Harnessing Common Features

Central to the MultiTaskModel is the shared bottom layer, comprising a sequence of fully connected layers with ReLU activations. Starting with an input layer of size equal to the number of features (233), it progresses to a hidden layer with 128 neurons, followed by another contraction to 64 neurons. This sequential downscaling captures a hierarchy of features at varying levels of abstraction, enabling the model to learn representations that are informative for both price prediction and category classification.

3.3 Task-Specific Top Layers: Tailored Outputs for Regression and Classification

Above the shared foundational layers, the model diverges into distinct pathways, each dedicated to one aspect of the prediction duo. The regression task is handled by a single-neuron output layer that maps the learned features to a house price, while the classification task leverages a separate output layer with a neuron count equal to the number of house categories. Both layers are critical in refining the model's outputs to address the nuanced objectives of the respective tasks.

3.4 Implementation with PyTorch Lightning: Streamlining Complex Workflows

Leveraging PyTorch Lightning's advanced capabilities, the model's construction and management are both streamlined. The framework offers a succinct, high-level interface for routine tasks, such as configuring optimizers and logging losses during training and validation. It simplifies the intricate process of multi-task learning, allowing for a focus on strategic design decisions and performance optimization.

The training step is defined to compute and combine the losses from both tasks, employing mean squared error for regression and cross-entropy for classification. This approach ensures that the gradient updates are influenced by the learning signals from both tasks. Validation steps further allow for the assessment of the model's performance, providing valuable feedback for refinement. The end-to-end workflow, from data loading with PyTorch's DataLoader to model fitting with Lightning's Trainer class, encapsulates the essence of efficient and scalable multi-task learning.

4. Experiments with Different Activation Functions and Optimizers (Model 1)

A crucial aspect of this multi-task learning project is the assessment of different activation functions and optimizers to identify their impacts on model performance. Through methodical testing with ReLU, Leaky ReLU, and PReLU activation functions, coupled with various learning rates, the experiment sought to find the most effective combination for both regression and classification tasks in our multi-task model.

4.1. Exploration of Activation Functions and Learning Rates

Initial experiments with the ReLU function established a performance baseline, which served as a point of comparison for alternative activation functions. The introduction of Leaky ReLU slightly improved model performance, allowing for continuous learning during backpropagation by permitting a small gradient when the unit's activation is below zero. PReLU further refined the model's capability by introducing learnable parameters, thus offering a dynamic adaptation during training, which proved to be beneficial, particularly in handling the classification task where nuanced differences in feature activation patterns are critical.

4.2. Mode Post – Experiments Evaluation and Performance Analysis

The multi-task learning model's performance was assessed through various metrics and visualizations post-experimentation. The regression aspect of the model demonstrated promising results, with a Mean Squared Error (MSE) of 0.1766, indicating that on average, the squared difference between the predicted and actual house prices is relatively low. This is further substantiated by the Root Mean Squared Error (RMSE) of 0.4203, which, being closer to zero, reflects accuracy in the model's predictive ability. The Mean Absolute Error (MAE) of 0.2993 reinforces the model's precision, denoting that the average absolute deviation from the actual prices is under control. The scatter plot of actual versus predicted prices (Figure 3) displays a concentration of data points along the line of perfect fit, albeit with some variance, underscoring the model's effectiveness yet hinting at the potential for further refinement.

The classification component's effectiveness (Figure 5) is captured in the classification report heatmap, which provides a deeper look at precision, recall, and F1-scores across different categories. While several categories exhibit exceptional F1-scores of 1.00, signifying perfect precision and recall, others, particularly category 4 and category 35, show a precision and recall of 0.00, indicating areas where the model fails to correctly identify the house category. These disparities suggest that while the model is adept at classifying certain categories, it struggles with others, possibly due to class imbalances or insufficient distinguishing features.

Combining these insights, it is evident that the model exhibits a robust capability in predicting house prices and categories to a considerable extent. However, the variation in classification performance across different categories points to the need for further model optimization, potentially through advanced feature engineering, class balancing techniques, or tailored adjustments in the learning algorithm.

5. Loss Functions and Advanced PyTorch Lightning Features (Model 2)

5.1. Unified Loss Function Implementation

In the MultiTaskModel 2, the regression component employs MSE as the loss function to quantify the average squared difference between the estimated values and the actual price, ensuring a robust penalty for large errors. The classification task utilizes Cross-Entropy Loss to measure the performance of the model's categorization by penalizing incorrect class predictions based on the confidence of the prediction. These two loss functions are harmoniously combined with a balanced weighting strategy, specified by the **loss_weights** parameter, ensuring neither task is favored, thus enabling the model to learn both tasks effectively without compromise.

5.2. Leveraging Advanced Features for Optimization and Evaluation

PyTorch Lightning's advanced features significantly streamline the training and evaluation process. Through the **TensorBoardLogger**, it is possible to gain valuable insights into the model's training dynamics, allowing for continuous monitoring and comparison of experiments. The **ModelCheckpoint** callback systematically preserves the best performing model based on validation loss, while the **EarlyStopping** callback preemptively halts training upon identifying stagnation in the reduction of regression loss, thereby preventing overfitting. This intelligent interplay of features ensures that the model reaches its peak performance without unnecessary computation.

5.3. Model Evaluation

5.3.1. Regression Performance

The model's regression capability was assessed using MSE, RMSE, and MAE. The reported MSE of 0.163 highlights the model's precision in price prediction, with lower values indicating higher accuracy. The RMSE value of 0.404, the square root of MSE, represents the standard deviation of the residuals and reflects the model's error in terms of actual price units. Meanwhile, the MAE of 0.246 provides a linear context to the average error magnitude, emphasizing the model's practical reliability.

5.3.2. Classification Efficacy

The classification performance is encapsulated by a heatmap (Figure 9) visualizing precision, recall, and F1-score across various house categories. The heatmap reveals that the model demonstrates exceptional precision and recall in several categories, with F1-scores frequently nearing 1.00, suggesting a high harmonic mean of precision and recall. This implies that the model not only correctly identifies the majority of the house categories but also maintains a low false positive rate, affirming its acumen in discernment.

The scatter plot of actual versus predicted prices (Figure 7) displays a high concentration of data points along the line of perfect prediction, indicating strong alignment between the model's predictions and the true values. The histogram of residuals (Figure 8), depicting the

distribution of prediction errors, further confirms the model's accuracy, with the bulk of residuals clustering near zero. Both visualizations corroborate the numerical metrics, illustrating the model's effectiveness in providing reliable and accurate house price estimations and classifications post-experimentation of advanced PyTorch features.

The evaluation results, underpinned by both the numerical metrics and their corresponding visualizations, manifest the successful calibration of the MultiTaskModel2. With adept utilization of advanced PyTorch Lightning features, the model not only provides high accuracy in its predictions but also shows remarkable performance in a multi-class classification setting, achieving an overall accuracy of 0.85. This balanced approach to multi-task learning signifies a significant stride in the predictive modeling of housing data.

6. Hyperparameter Tuning with Optuna

The pursuit of the optimal configuration for the MultiTaskModel was conducted through Optuna, a hyperparameter optimization framework. The **objective** function guided the search, with the learning rate (**lr**) explored over a logarithmic scale from 1e-5 to 1e-1. The architecture's depth and breadth were varied, with the number of layers (**n_layers**) ranging from 1 to 3 and the size of each layer (**n_units_l{i}**) chosen from a set of 64, 128, or 256 units.

During the trials, each potential set of parameters was instantiated into a MultiTaskModel and evaluated using PyTorch Lightning's Trainer, which was configured with a TensorBoardLogger for detailed logging, a ModelCheckpoint callback to save the best-performing models, and a custom early stopping callback to prevent overfitting by halting the training once the validation loss ceased to decrease for a patience of three epochs.

After rigorous experimentation, spanning 10 trials, the study identified the most effective combination of hyperparameters. The learning rate was fine-tuned to approximately 0.0028545627033560875. Concurrently, the optimal structure consisted of three layers with the respective sizes of 128, 128, and 64 units. This configuration not only minimized the validation loss but also culminated in the best overall model performance in terms of both regression and classification tasks.

This methodical approach to hyperparameter tuning underscored the significance of precision in model configuration, where even slight alterations to parameters such as learning rate and layer sizes can markedly influence the model's predictive prowess. The final model, equipped with these optimized parameters, demonstrated superior performance, offering a robust solution for the simultaneous regression and classification challenges posed by the dataset.

6.1. Final Model Evaluation and Performance Analysis

The final model, fine-tuned through the Optuna framework, exhibits a notable performance enhancement when compared to the initial versions. The MSE settled at a value of 0.1549, reflecting a modest decrease from the previously recorded 0.1639, indicating an

improvement in the model's accuracy in predicting house prices. This is supported by the accompanying RMSE and MAE which stand at 0.3936 and 0.234, respectively. The reduction in these metrics suggests a closer alignment of the predicted values to the actual prices, and a smaller average deviation from the true values.

Visual evaluations, such as the 'Actual vs. Predicted Prices' scatter plot (Figure 11) and the 'Histogram of Residuals' (Figure 12), provide a graphical representation of the model's predictive accuracy. The scatter plot reveals a dense alignment along the line of perfect prediction, while the histogram shows the residuals clustered around zero, with a fairly symmetrical distribution—both of which are indicators of a well-fitting model.

In terms of classification, the heatmap (Figure 13) from the classification report indicates varying degrees of precision, recall, and F1 score across different categories. The precision scores are high for several classes, denoting a low false positive rate, while perfect recall scores in certain classes indicate no false negatives. The F1 scores, which balance precision and recall, are predominantly high, suggesting a harmonious balance between these metrics for most classes.

The overall accuracy reached an admirable 0.88, surpassing the earlier model's accuracy of 0.85. This improvement reflects the model's enhanced ability to correctly classify houses into the appropriate categories. The advanced features of PyTorch Lightning, including efficient data handling, and the combined regression and classification loss function, have evidently contributed to these refined outcomes.

In summary, the iterative process of hyperparameter tuning, along with the thoughtful integration of advanced training features, has culminated in a robust model that exhibits commendable precision in its predictions. The visual and quantitative analyses of the model’s performance collectively demonstrate its refined predictive capabilities and its adeptness at both regression and classification tasks within the multi-task learning paradigm.

7. Discussion and Comparative Analysis

The evolution of the multi-task learning model through various stages of enhancement has led to significant performance improvements. The first model, utilizing ReLU activation and a learning rate of 0.001, laid a solid foundation, evidenced by its MSE of 0.1766, RMSE of 0.4203, and MAE of 0.2993, demonstrating a capable but improvable prediction capability.

The second model incorporated advanced PyTorch Lightning features, achieving a slightly better MSE of 0.1639, with corresponding improvements in RMSE and MAE. This model benefited from tailored features such as logging and checkpointing, yielding an overall accuracy of 0.85, showing a marked progression in classification precision.

However, the third model, which underwent hyperparameter tuning with Optuna, outperformed its predecessors, showcasing the lowest MSE of 0.1549 and corresponding

decreases in RMSE and MAE. The precision of predictions, as depicted by the scatter plot and histogram of residuals, showed tighter clustering around the line of best fit and a more symmetrical residual distribution. Furthermore, classification accuracy peaked at 0.88, illustrating the fine-tuning impact of Optuna in model optimization.

7.1. Model Strengths and Limitations

The final model's strength lies in its refined calibration, where the hyperparameter tuning has been adeptly leveraged to enhance both regression and classification accuracy. The high degree of precision, coupled with advanced Lightning features, has allowed the model to achieve high efficacy in predicting diverse house categories.

Despite these strengths, limitations persist. The model's predictive performance, while high, still shows room for improvement in certain classes where precision and recall are less than ideal. This indicates potential class imbalance or feature underrepresentation, suggesting a need for further optimization.

7.2. Challenges Faced

- **Class Imbalance:** Some categories in the classification task have a very low number of instances, leading to a lack of training data for those categories and potentially causing the model to perform poorly on them. This is evident in categories with a precision or recall of 0.00, indicating that the model is unable to correctly identify instances of these less represented categories.
- **Complex Feature Relationships:** The complexity of the relationships between features and the house prices and categories may not be fully captured by the model. For instance, nonlinear relationships or interactions between features could require more advanced modeling techniques or feature engineering to understand fully.
- **Overfitting/Underfitting Trade-off:** Adjusting the model's capacity and regularization to balance the trade-off between overfitting and underfitting is an ongoing challenge. The ideal model should generalize well to unseen data without losing the ability to capture the underlying patterns in the training data.
- **Loss Function Balance:** The model uses a combined loss function to train on both regression and classification tasks simultaneously. Balancing these can be delicate, as giving too much weight to one can undermine the performance of the other.

7.3. Further Improvements

To address the challenges identified in the final model, a suite of strategic interventions could be enacted. For class imbalances, techniques such as resampling or applying synthetic data generation like SMOTE could level the playing field for minority classes. In tackling the intricacies of complex feature relationships, enhanced feature engineering, and the deployment of advanced modeling techniques capable of capturing nuanced patterns would be beneficial. When confronting the delicate balance between overfitting and underfitting, methods such as

cross-validation and regularization can offer a safeguard, ensuring the model's generalizability. Lastly, fine-tuning the combined loss function could better harmonize the regression and classification tasks, potentially improving the overall efficacy of the model. These solutions, when implemented thoughtfully, could significantly bolster the model's predictive prowess while maintaining a vigilant stance against introducing new issues.

In conclusion, while the third model achieved the best performance, demonstrating the efficacy of systematic hyperparameter optimization, it is not without the need for further improvements. The machine learning journey is iterative, and each challenge surmounted provides insight into potential enhancements. The continued refinement of the model could focus on addressing the specific weaknesses identified during evaluation to achieve even higher accuracy and precision in its predictions.

8. Conclusion

This project's journey in creating a multi-task learning model has been marked by significant learning and advancements. Starting with a basic model employing ReLU and evolving through the incorporation of advanced PyTorch Lightning features, to finally refining the model using Optuna for hyperparameter tuning, there have been consistent improvements in accuracy and classification capabilities.

The final model showcases the power of hyperparameter optimization and the strength of multi-task learning, standing out with its high accuracy and precision. Despite its successes, opportunities for enhancement remain, particularly in addressing class imbalances and capturing complex feature relationships. These challenges provide clear pathways for future work, where methods like resampling and advanced feature engineering could further improve model performance.

Reflecting on the project, it is evident that the iterative nature of machine learning allows for continuous improvement. The insights gained here set a foundation for future models to build upon, promising even more sophisticated and accurate predictive tools in the field of real estate analytics.

9. Appendix

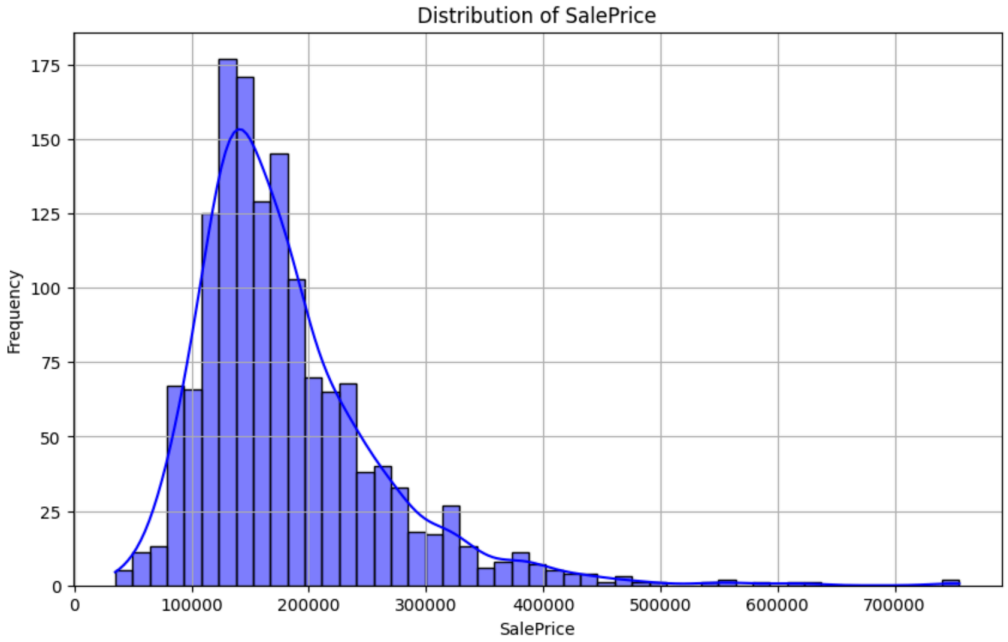

Figure 1: Distribution of Sales Price

Figure 2: Regression Performance Metrics on Model after Experiments

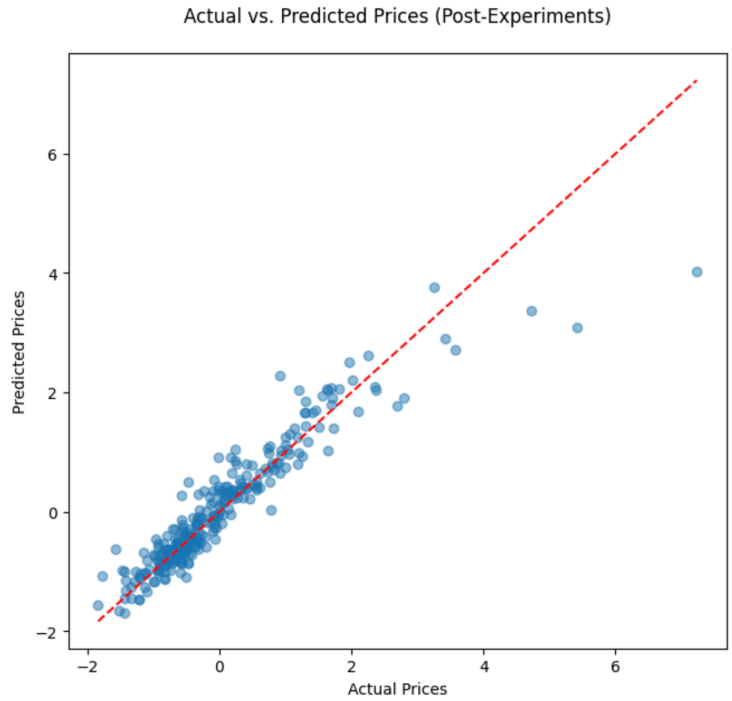

Figure 3: Scatter plot for actual vs predicted prices for Model after Experiments

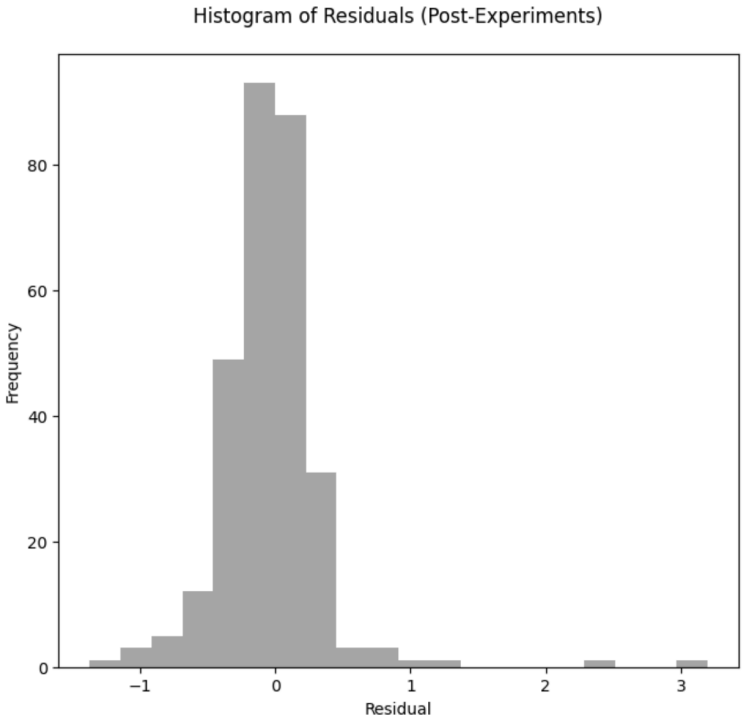

Figure 4: Histogram of the Residuals of Model after Experiments

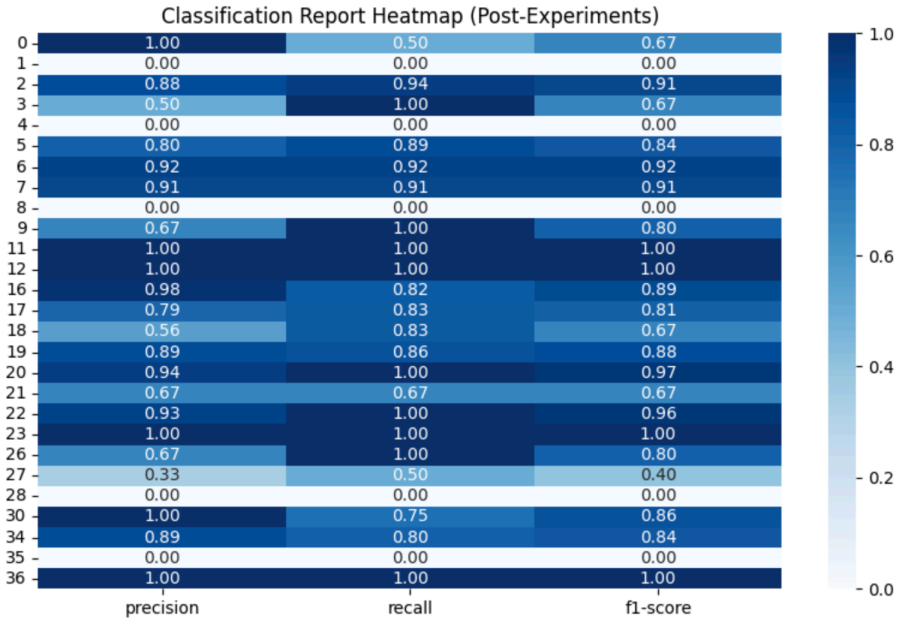

Figure 5: Classification Report Heatmap on Model after Experiments

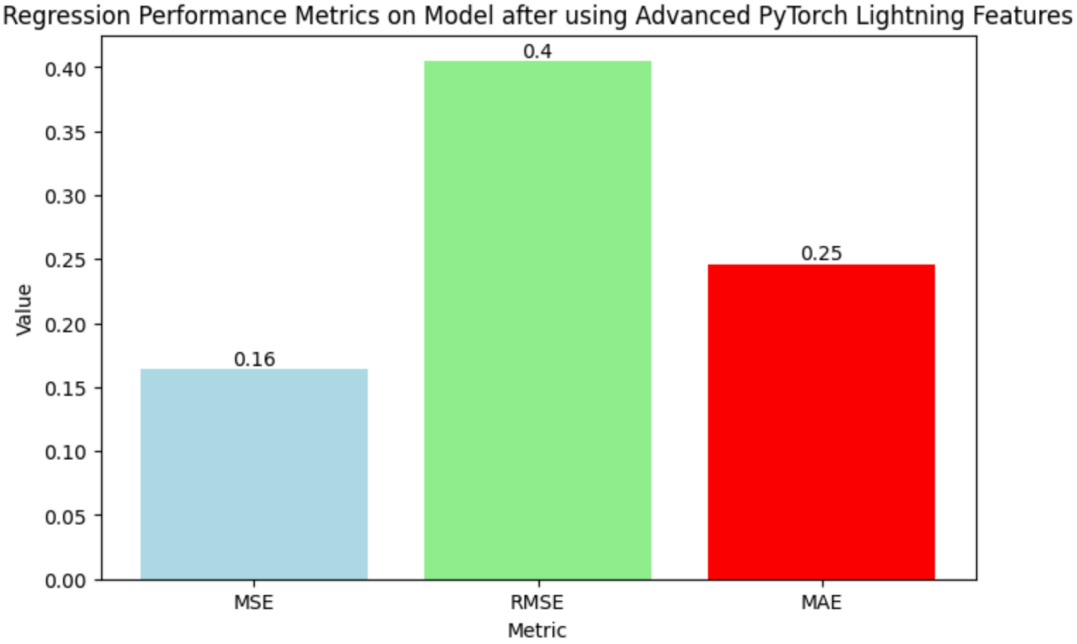

Figure 6: Regression Performance Metrics on Model after Using Advanced PyTorch Lightning Features

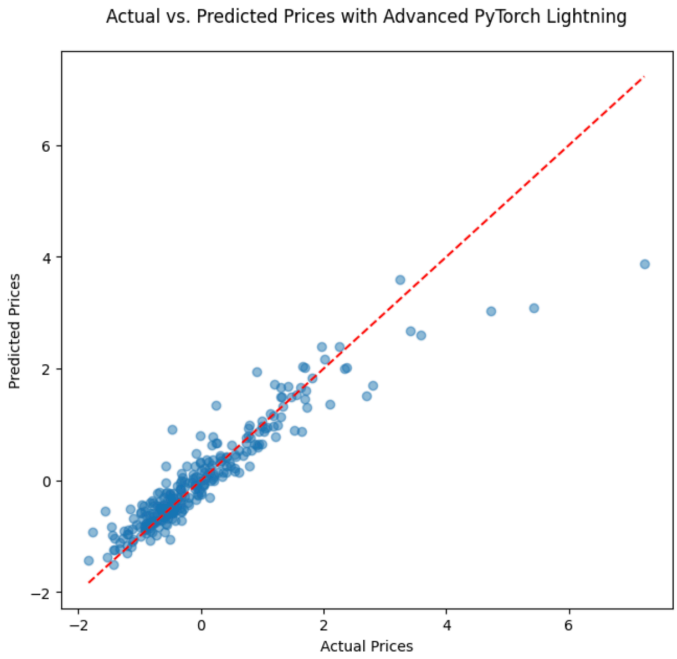

Figure 7: Scatter plot for actual vs predicted prices for Model after Using Advanced PyTorch Lightning Features

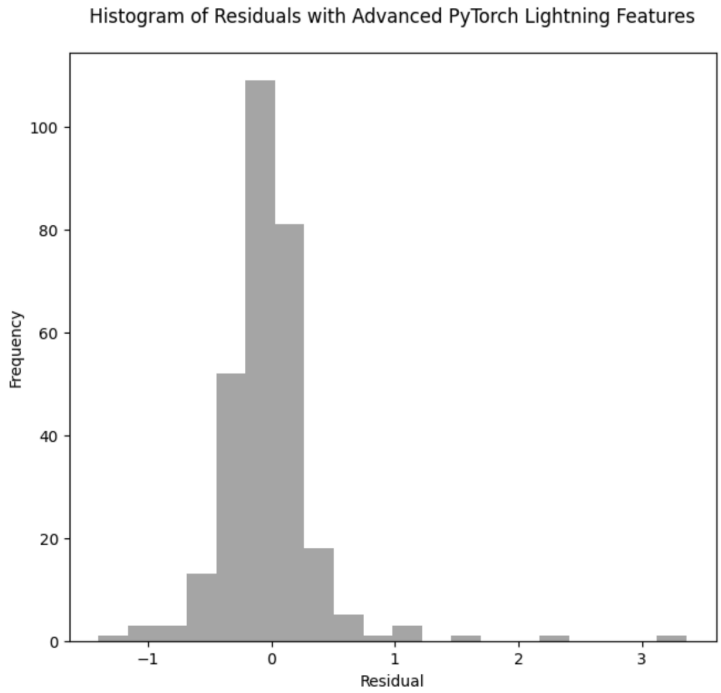

Figure 8: Histogram of the Residuals of Model after Using Advanced PyTorch Lightning Features

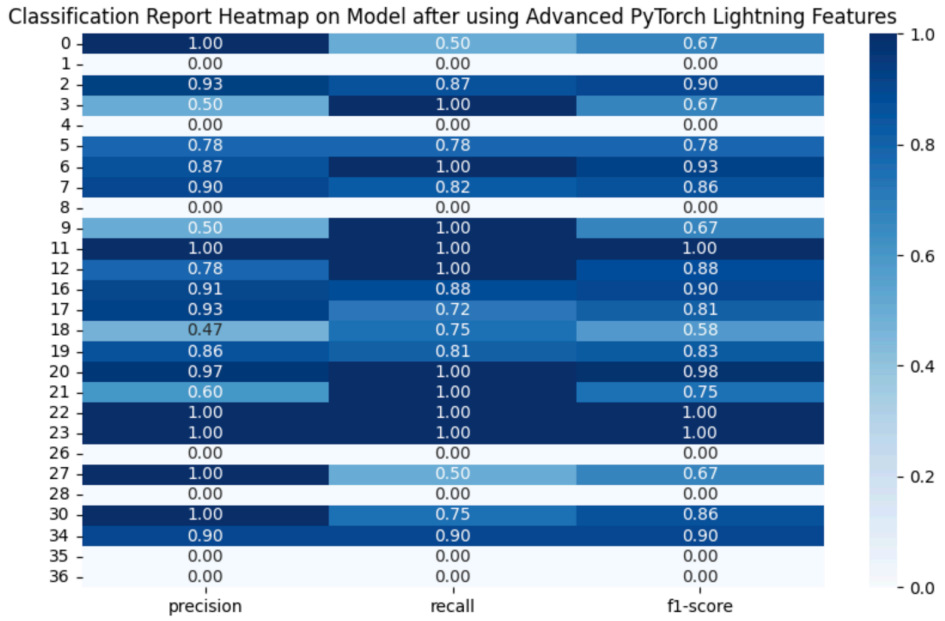

Figure 9: Classification Report Heatmap on Model after Using Advanced PyTorch Lightning Features

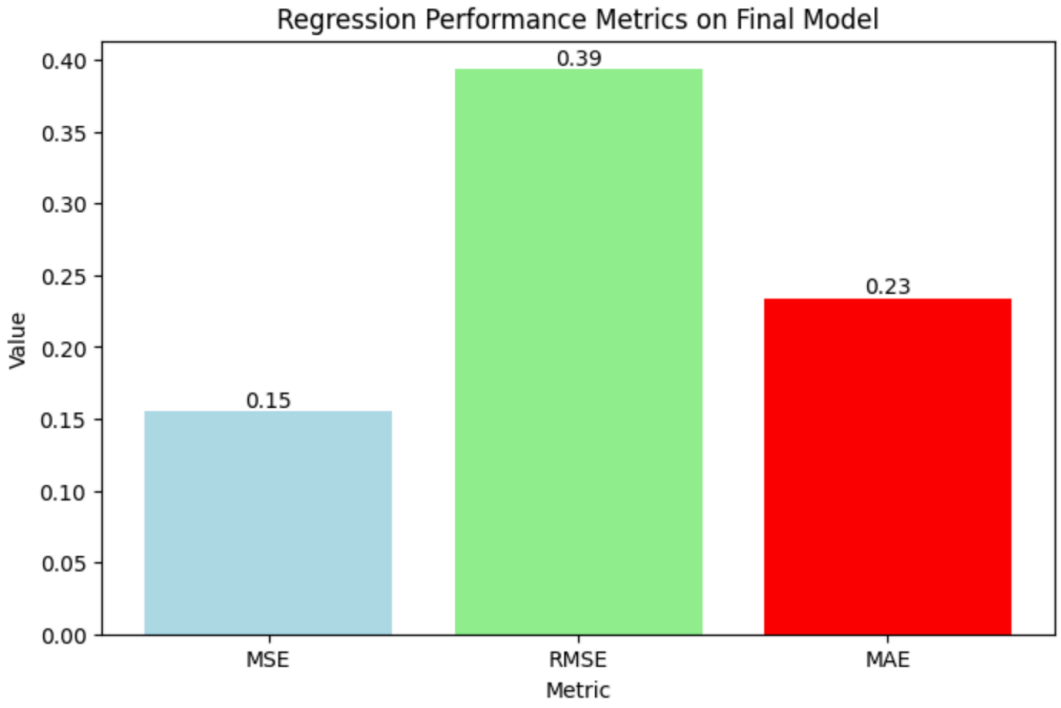

Figure 10: Regression Performance Metrics on Final Model

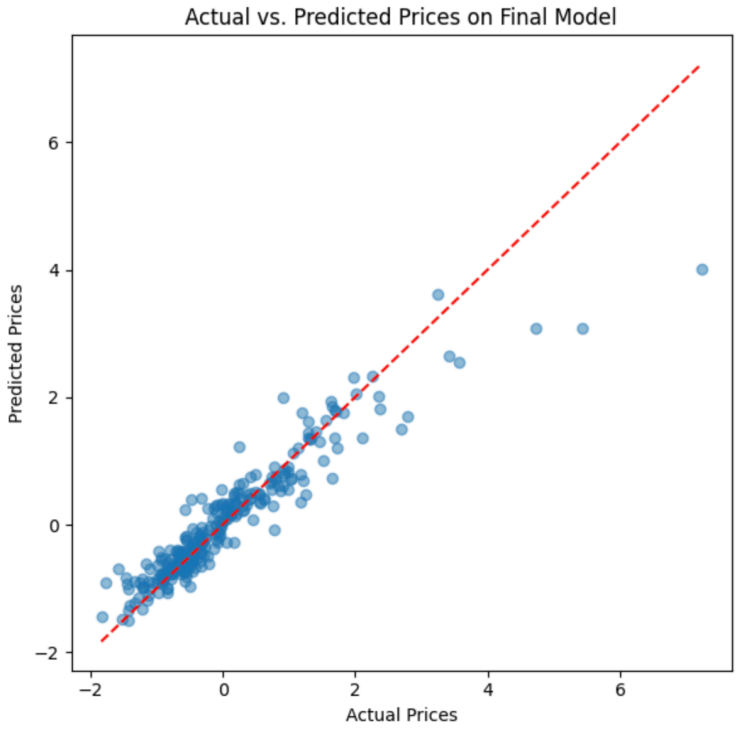

Figure 11: Scatter plot for actual vs predicted prices for Final Model

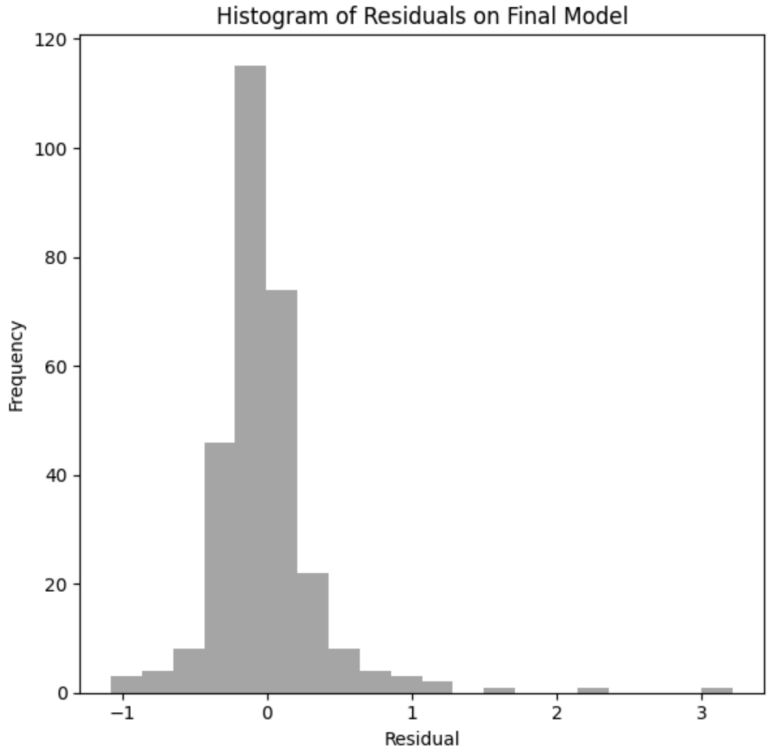

Figure 12: Histogram of the Residuals of Final Model

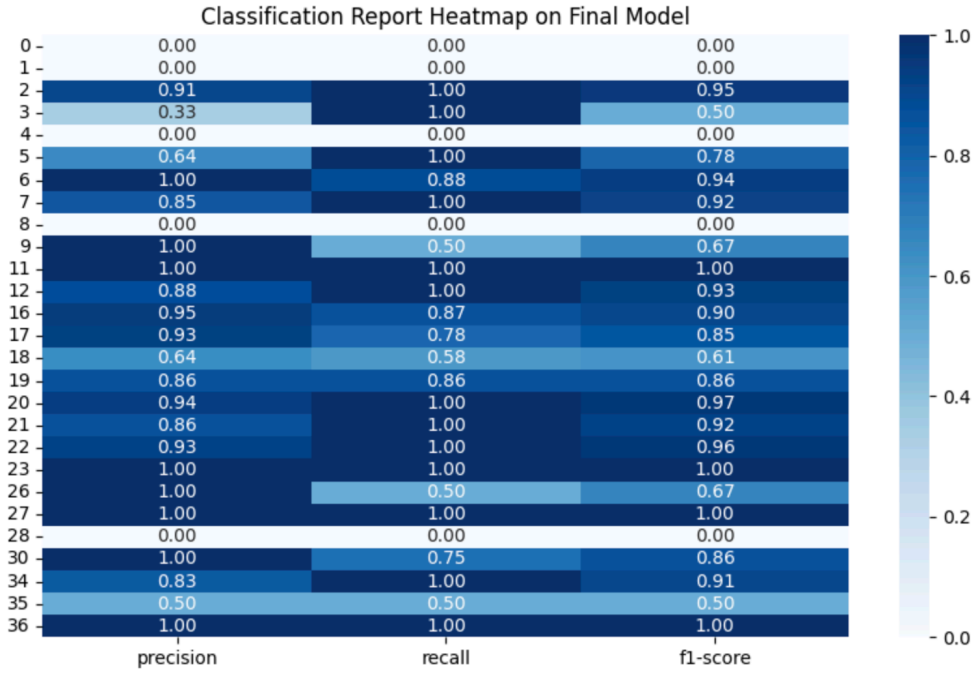

Figure 13: Classification Report Heatmap on Final Model