Linear Regression and Regularization Techniques on Boston Housing Dataset

Mark Josh N. Alvear
College of Information and Computing Sciences
University of Santo Tomas
Manila, Philippines
markjosh.alvear.cics@ust.edu.ph

Abstract – This paper analyzes housing prices in Boston using the Boston Housing Dataset to predict the median value of owner-occupied homes (MEDV). Linear regression and its regularized variants: Ridge, Lasso, and Elastic Net are applied after exploratory data analysis, feature selection, and preprocessing, including log transformations and outlier removal. Regularized models, particularly Elastic Net, outperform the baseline, reducing overfitting and improving generalizability. Key features like LSTAT, RM, and DIS significantly influence housing prices. The study provides practical insights for housing market analysis, emphasizing the importance of regularization and feature selection in predictive modeling.

I. INTRODUCTION

The Boston Housing Dataset, a staple in regression analysis, was originally compiled by the U.S. Census Service in 1978 and later archived by StatLib. This dataset includes the median house prices in Boston, Massachusetts, along with various factors influencing these prices, such as the average number of rooms, proximity to radial highways, and crime rates. Notably, it also includes a variable representing the proportion of Black residents per town, which has sparked ethical concerns regarding potential biases in predictive modeling. Given the dataset's age and ethical considerations, its ability to accurately forecast modern housing prices is limited. To address these challenges, this study aims to develop a straightforward and interpretable model using linear regression as a baseline. Linear regression is chosen for its simplicity and clarity in illustrating the relationships between features and house prices, making it an ideal starting point for beginners in machine learning. Building on this baseline, the model will undergo a thorough evaluation process. Regularization techniques such as Lasso, Ridge, and Elastic Net will be applied to enhance model performance. The model's accuracy and effectiveness in predicting house prices will be assessed using statistical metrics like R-squared and Mean Squared Error (MSE). This comprehensive evaluation will also explore the impact of each regularization technique on model performance, offering a detailed understanding of the model's capabilities.

A. Objectives

- **Design a robust model:** Develop a predictive model using the Boston Housing Dataset, ensuring it avoids overfitting or underfitting. Aim for an MSE within an acceptable range and an R-squared value close to 1.
- **Identify key features:** Analyze model coefficients to determine the most influential features in predicting house prices, enhancing understanding of feature contributions.
- **Ensure model robustness:** Use cross-validation to evaluate model performance across different data subsets, ensuring consistency and reliability while minimizing overfitting risks.

II. METHODOLOGY

This section outlines the methodology for designing and evaluating regression models to predict housing prices using the dataset, covering data preprocessing, model implementation, and performance evaluation.

A. Data Preprocessing

Exploratory Data Analysis (EDA) involves key steps to understand the dataset, such as handling missing data, visualizing data distributions, and addressing outliers. Scatterplots and heatmaps will be used to examine variable relationships and detect high correlations. Data normalization, using techniques like StandardScaler, will ensure all features contribute equally to the model, enhancing predictive capabilities.

B. Data Modeling

1. Linear Regression

This foundational model offers a clear mathematical formula for predictions and is widely applicable. Key assumptions, such as linearity and normality of residuals, will be checked to ensure model validity.

2. Lasso Regression

This technique adds a penalty to the model to address overfitting and perform feature selection, simplifying the model and improving interpretability.

3. Ridge Regression

By adding a penalty for large coefficients, Ridge regression reduces model complexity and enhances generalization to new data. Elastic Net Regression: Combining Lasso and Ridge penalties, Elastic Net handles situations with many or highly correlated features, optimizing model performance through cross-validation.

4. Hyperparameter Tuning:

Although grid search is not used, hyperparameter tuning remains crucial for optimizing model settings and enhancing performance. Model Evaluation: R-squared and MSE: These metrics will assess model fit and prediction accuracy, with higher R-squared and lower MSE indicating better performance. Training vs. Test Data: Comparing model performance on training and test data will help identify overfitting, underfitting, or generalization issues.

III. DATA ANALYSIS

The Boston Housing Dataset consists of 506 rows and 14 columns, each representing different attributes that potentially influence housing prices. Below is an analysis of the dataset's key features and their initial exploration.

A. Data Overview

- **CRIM:** Per capita crime rate by town.
- **ZN:** Proportion of residential land zoned for lots over 25,000 sq. ft.
- **INDUS:** Proportion of non-retail business acres per town.
- **CHAS:** Charles River dummy variable (1 if tract bounds river; 0 otherwise).
- **NOX:** Nitric oxides concentration (parts per 10 million).
- **RM**: Average number of rooms per dwelling.
- **AGE**: Proportion of owner-occupied units built prior to 1940.
- **DIS:** Weighted distances to five Boston employment centers.
- **RAD:** Index of accessibility to radial highways.
- **TAX**: Full-value property tax rate per $10,000.
- **PTRATIO:** Pupil-teacher ratio by town.
- **B:** 1000(Bk - 0.63)^2 where Bk is the proportion of Black residents by town.
- **LSTAT:** Percentage of lower status of the population.
- **MEDV:** Median value of owner-occupied homes in $1000s.

B. Exploratory Data Analysis(EDA)

This part analyzes the dataset to see any relationships between the features. It will also show patterns, correlations and issues of the dataset.

1. Descriptive Statistics

Statistical summary of the dataset:

	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

	AGE	DIS	RAD	TAX	PTRATIO	B
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032
std	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000
25%	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500
50%	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000
75%	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000

	LSTAT	MEDV
count	506.000000	506.000000
mean	12.653063	22.532806
std	7.141062	9.197104
min	1.730000	5.000000
25%	6.950000	17.025000
50%	11.360000	21.200000
75%	16.955000	25.000000
max	37.970000	50.000000

Figure 1: Statistical Summary of Dataset

This table shows descriptive statistics for the Boston Housing dataset, including count, mean,

standard deviation, minimum, quartiles (25%, 50%, 75%), and maximum values for each variable. It summarizes the distribution and range of values for each feature.

```
Missing values in each column:
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

Figure 1.1: Checking Null Values of each Column

*This figure confirms the Boston Housing dataset contains **no missing values**. Each feature has a count of 0 for missing values, indicating a complete dataset.*

2. *Data Visualization*

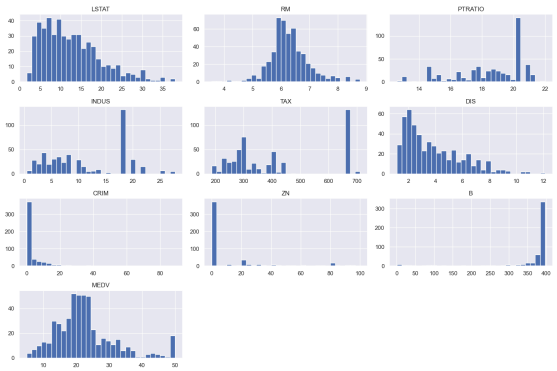

Figure 2: Histograms of selected features before transformation

Figure 2 shows histograms of several features from the Boston Housing dataset before any transformations were applied. It visualizes the distribution of each feature, revealing their ranges, central tendencies, and potential skewness. For example, 'LSTAT' appears right-skewed, while 'RM' roughly approximates a normal distribution. These distributions are important to consider for feature engineering and model selection.

3. *Correlation Analysis*

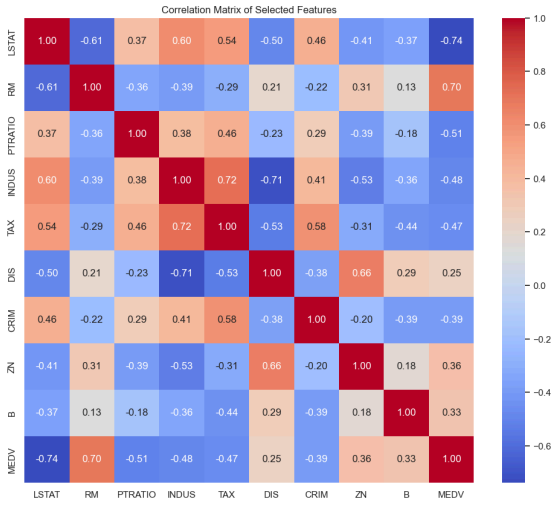

Figure 3: Correlation Matrix of Selected Features

Figure 3 displays the correlation matrix of selected features from the Boston Housing dataset, revealing pairwise relationships between variables like LSTAT, RM, and MEDV. Color intensity and numbers indicate correlation strength and direction.

Why these selected features? These features, encompassing socioeconomic factors (LSTAT), property attributes (RM, ZN), location characteristics (DIS), and other relevant factors (CRIM, TAX, INDUS, PTRATIO, B), are commonly used in housing price prediction models. They represent a range of influences on housing values, including socioeconomic status, property size, accessibility, tax rates, and environmental factors. MEDV is the target variable (median home value).

IV. DATA PREPROCESSING

```
Skewness of selected features before transformation:
LSTAT      0.906460
RM         0.403612
PTRATIO   -0.802325
INDUS      0.295022
TAX        0.669956
DIS        1.011781
CRIM       5.223149
ZN         2.225666
B         -2.890374
dtype: float64
```

Figure 4: Skewness of Selected Features before Transformation

This output displays the skewness of features in the Boston Housing dataset *before* any transformations are applied. Positive skewness values indicate a right-skewed distribution, where the tail of the distribution extends more

towards higher values. Specifically, CRIM (crime rate) shows a very high positive skewness (5.22), suggesting a large concentration of low values with some extreme high values. LSTAT (lower status of the population) and DIS (distances to employment centers) also exhibit moderate positive skewness, indicating similar, though less extreme, distribution patterns. These skewness values justify the need for data transformation, such as a log transformation, to improve the symmetry of these features' distributions, which can be beneficial for many machine learning models. The listed skewness for the other features will likely be evaluated to determine if they also require transformation.

```
Skewness of selected features after transformation:
LSTAT     -0.187195
RM         0.403612
PTRATIO   -0.802325
INDUS      0.295022
TAX        0.669956
DIS        0.331561
CRIM       1.269201
ZN         2.225666
B         -2.890374
dtype: float64
```

Figure 4.1 Skewness of selected features after transformation

This output shows the skewness of selected features from the Boston Housing dataset *after* a transformation, likely intended to reduce skew. Ideally, skewness should be near zero for a symmetrical distribution. While some features (LSTAT, DIS, INDUS) show reduced skewness, suggesting the transformation was effective, others (RM, TAX, CRIM, ZN, B) still exhibit moderate to high skewness, indicating the transformation may not have fully addressed their distributional issues. PTRATIO now has a moderate negative skew. These persistent skewness values suggest further investigation is needed, potentially involving different transformations or outlier handling, as skewed features can negatively impact model performance and the validity of some statistical assumptions in machine learning. **NOTE**: CRIM was log-transformed. LSTAT was log-transformed. DIS was log-transformed.

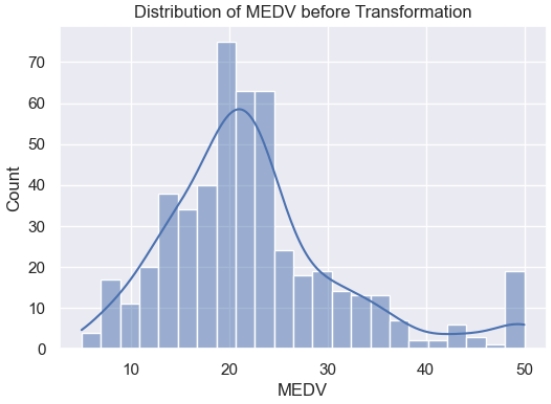

Figure 4.2 Distribution of MEDV before Transformation

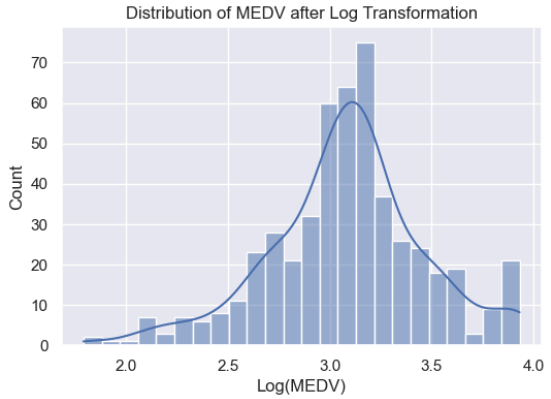

Figure 4.3 Distribution of MEDV after Transformation

The figures above displays the distribution of the 'MEDV' (median value of owner-occupied homes) variable from the Boston Housing dataset before and after a logarithmic transformation. The "before" histogram shows a positively skewed distribution, with a long tail extending towards higher values. This skewness is addressed by the log transformation, and the "after" histogram demonstrates a more symmetrical, near-normal distribution. Log transformation is often applied to reduce skewness and improve the performance of statistical models that assume normality or benefit from more symmetrical data.

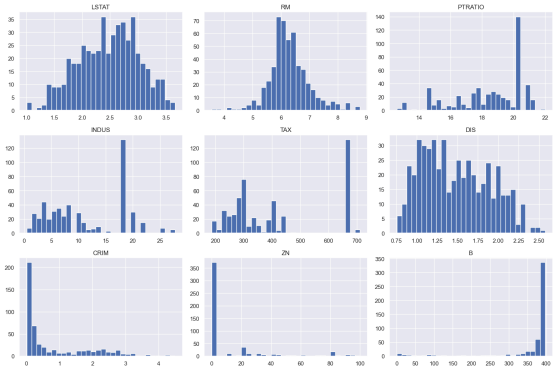

Figure 4.4 Histograms of selected features after transformation:

Figure 4.4 displays histograms of selected features from the Boston Housing dataset after a transformation, intended to reduce skewness. While some features (LSTAT, DIS) show improved symmetry, others (RM, TAX, CRIM, ZN, B) still exhibit significant skewness, indicating the transformation may not have been fully effective. PTRATIO appears to have developed a negative skew. These results suggest that further data exploration and potential alternative transformations or outlier handling may be necessary to address the remaining skewness issues and improve the performance of machine learning models.

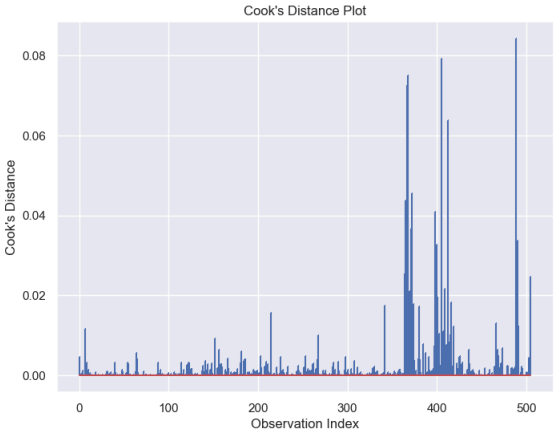

Figure 4.5 Cook's Distance Plot

The output shows a Cook's Distance plot and confirms the removal of 34 influential data points. Cook's Distance measures the influence of each observation on the regression model; high values indicate points that unduly affect the model's coefficients. Removing these influential outliers, justified by their potential to skew results and inflate standard errors, aims to create a more robust and generalizable model. The plot visually highlights these influential points, allowing for assessment of their impact.

1. *Feature Scaling*

	LSTAT	RM	PTRATIO	INDUS	TAX	DIS	CRIM	ZN	B
0	1.788421	6.575	15.3	2.31	296.0	1.627278	0.006300	18.0	396.90
1	2.316488	6.421	17.8	7.07	242.0	1.786261	0.026944	0.0	396.90
2	1.615420	7.185	17.8	7.07	242.0	1.786261	0.026924	0.0	392.83
3	1.371181	6.998	18.7	2.18	222.0	1.954757	0.031857	0.0	394.63
4	1.845300	7.147	18.7	2.18	222.0	1.954757	0.066770	0.0	396.90
...	...	...	...	...	...	...	...	...	...
467	2.729812	6.027	19.2	9.69	391.0	1.252249	0.202435	0.0	396.90
468	2.367436	6.593	21.0	11.93	273.0	1.246630	0.060747	0.0	391.99
469	2.310553	6.120	21.0	11.93	273.0	1.190127	0.044275	0.0	396.90
470	1.893112	6.976	21.0	11.93	273.0	1.152943	0.058986	0.0	396.90
471	2.012233	6.794	21.0	11.93	273.0	1.220505	0.103991	0.0	393.45

472 rows × 9 columns

Figure 5: Scaling the Features

The output shows a portion of the `X_clean` DataFrame. This DataFrame contains the feature values *after* outlier removal but *before* feature scaling. The values for each feature (LSTAT, RM, PTRATIO, INDUS, TAX, DIS, CRIM, ZN, B) are displayed for a subset of the observations. Note that these values are on their original scales, which can vary widely between features.

Variance Inflation Factors (VIF) after handling outliers:

	Features	VIF
0	LSTAT	3.429818
1	RM	2.289173
2	PTRATIO	1.553230
3	INDUS	3.356531
4	TAX	5.228403
5	DIS	3.357493
6	CRIM	5.542614
7	ZN	2.028650
8	B	1.434527

Figure 5.1: Variance Inflation Factors (VIF) after handling outliers

This output displays the Variance Inflation Factors (VIFs) for a set of features after outliers have been handled. VIF measures multicollinearity, or the correlation between predictor variables in a regression model. High multicollinearity can lead to unstable coefficient estimates and inflated standard errors, making it difficult to interpret the individual effects of predictors. A VIF of 1 indicates no multicollinearity, while values above 5 or 10 are often considered problematic. Here, TAX (5.23) and CRIM (5.54) have VIFs slightly above 5, suggesting moderate multicollinearity persists even after outlier treatment. All other features have VIFs below 5. Because outliers can distort

multicollinearity assessments, VIF is re-evaluated after outlier handling. Addressing multicollinearity, potentially through feature removal or combination, can improve model reliability and interpretability.

2. *Feature Engineering*

	LSTAT	RM	PTRATIO	INDUS	TAX	DIS	CRIM	ZN	B	RM_LSTAT	DIS_INDUS
0	-1.267930	0.398048	-1.422053	-1.243536	-0.606743	0.287313	-0.762819	0.246028	0.421783	-0.504697	-0.357284
1	-0.259059	0.163916	-0.267344	-0.536823	-0.942035	0.682547	-0.740910	-0.506268	0.421783	-0.042464	-0.366407
2	-1.598447	1.325454	-0.267344	-0.536823	-0.942035	0.682547	-0.740931	-0.506268	0.374791	-2.118668	-0.366407
3	-2.065065	1.041151	0.148351	-1.262837	-1.066217	1.101428	-0.735696	-0.506268	0.395574	-2.150045	-1.390925
4	-1.159261	1.267681	0.148351	-1.262837	-1.066217	1.101428	-0.698644	-0.506268	0.421783	-1.469574	-1.390925
...	...	...	...	...	...	...	...	...	...	...	...
467	0.530595	-0.435097	0.379293	-0.147834	-0.016878	-0.645013	-0.554669	-0.506268	0.421783	-0.230860	0.095355
468	-0.161722	0.425414	1.210683	0.184737	-0.749552	-0.658981	-0.705036	-0.506268	0.365092	-0.068799	-0.121738
469	-0.270397	-0.293705	1.210683	0.184737	-0.749552	-0.799447	-0.722517	-0.506268	0.421783	0.079417	-0.147688
470	-1.067917	1.007704	1.210683	0.184737	-0.749552	-0.891889	-0.706906	-0.506268	0.421783	-1.076144	-0.164765
471	-0.840337	0.731002	1.210683	0.184737	-0.749552	-0.723927	-0.659144	-0.506268	0.381949	-0.614288	-0.133736

472 rows × 11 columns

Figure 6:Creating Interaction Terms

Figure 6 demonstrates feature engineering through the creation of interaction terms. Starting with scaled features (likely standardized using z-score normalization), the code converts the scaled NumPy array back into a Pandas DataFrame for easier manipulation. It then generates two new interaction features: `RM_LSTAT` and `DIS_INDUS`. `RM_LSTAT` is created by multiplying the scaled values of `RM` (average number of rooms) and `LSTAT` (lower status of the population), hypothesizing that the impact of room number on house price varies with socioeconomic status. Similarly, `DIS_INDUS` is created by multiplying scaled `DIS` (distance to employment centers) and `INDUS` (proportion of industrial land), assuming the effect of distance to jobs depends on the industrial character of the area. These interaction terms capture potential combined effects of the original features. The feature list is updated to include these new terms, and the resulting DataFrame (`X_final`) now contains both the scaled original features and the engineered interaction terms, ready for model training. Creating interaction terms is a crucial step in feature engineering as it allows models to learn more complex relationships present in the data, potentially leading to improved predictive accuracy. Scaling ***before*** creating interactions is generally recommended to ensure all features, including the interaction terms, are on a similar scale, preventing features with larger original ranges from dominating the model.

3. *Splitting The Data*

```
# Splitting the Data
# Split the dataset into training and testing sets
X_train, X_test, y_train_log, y_test_log = train_test_split(
    X_final, y_clean_log, test_size=0.2, random_state=42)
```

Figure 7: Code Snippet of Splitting the Data

This code snippet engineers interaction terms from scaled features. It converts the scaled data back to a DataFrame, creates "RM_LSTAT" (rooms * lower status) and "DIS_INDUS" (distance to employment * industrial acres) interaction terms, updates the feature list, and creates the final feature matrix `X_final`. Scaling is done before interaction creation to ensure consistent scales across all features.

V. MODEL IMPLEMENTATION

In this study, we focus on implementing linear regression models to predict housing prices using the Boston Housing Dataset. Our approach involves using Linear Regression as a baseline and enhancing it with regularization techniques: Ridge, Lasso, and Elastic Net. These models are chosen for their ability to handle different aspects of data complexity and feature selection.

A. Linear Regression

Linear Regression serves as our foundational model due to its simplicity and interpretability. It provides a straightforward way to understand the relationship between features and the target variable, MEDV. This model is ideal for beginners and offers a clear baseline for comparison with more complex models.

B. Ridge Regression

Ridge Regression, or L2 regularization, is employed to address multicollinearity by adding a penalty for large coefficients. This helps prevent overfitting, especially when dealing with datasets that have many correlated features. Ridge Regression ensures that the model generalizes better to new data by reducing the impact of less important features.

C. Lasso Regression

Lasso Regression, or L1 regularization, introduces a penalty that can shrink some coefficients to zero, effectively performing feature selection. This is particularly useful in

high-dimensional datasets, where it helps simplify the model by eliminating irrelevant features, enhancing interpretability and reducing overfitting.

D. Elastic Net Regression

Elastic Net combines the penalties of both Ridge and Lasso, making it suitable for situations with many features or highly correlated variables. By balancing L1 and L2 penalties, Elastic Net can handle complex datasets more effectively, providing a robust model that benefits from both feature selection and multicollinearity management

E. Model Evaluation

Each model is evaluated using R-squared and Mean Squared Error (MSE) metrics to assess their accuracy and effectiveness. Cross-validation is employed to ensure model robustness and consistency across different data subsets, minimizing the risk of overfitting. This approach allows us to explore the strengths of each model type, providing a comprehensive understanding of their capabilities in predicting housing prices.

VI. EXPERIMENTS

The following experiments evaluate the predictive performance of several regression models on the Boston Housing dataset, now enhanced with engineered interaction terms and cleaned of outliers. We compare the baseline linear regression model with regularized variants (Ridge, Lasso, and Elastic Net) to assess the impact of regularization and feature selection on predictive accuracy.

1. Linear Regression

```python
# Instantiate and train the Linear Regression model
lr = LinearRegression()
lr.fit(X_train, y_train_log)

# Predict on test set
y_pred_log_lr = lr.predict(X_test)

# Convert predictions back to original scale
y_pred_lr = np.expm1(y_pred_log_lr)
y_test = np.expm1(y_test_log)

# Evaluate the model
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)
rmse_lr = np.sqrt(mse_lr)

print("\nLinear Regression Performance on Test Set:")
print(f"MSE: {mse_lr:.2f}")
print(f"R-squared: {r2_lr:.4f}")
```

Figure 8: Code Snippet of Linear Regression

The code snippet demonstrates the implementation of a Linear Regression model to predict housing prices. Initially, the model is instantiated and trained using the training dataset, where the target variable is log-transformed to stabilize variance. Predictions are then made on the test set, resulting in log-scaled predicted values. These predictions, along with the actual test values, are converted back to their original scale using the exponential function to reverse the log transformation. The model's performance is evaluated using Mean Squared Error (MSE) and R-squared (R^2) metrics. MSE measures the average squared difference between predicted and actual values, while R-squared indicates the proportion of variance in the target variable explained by the model. The results are then printed as it showcases the model's accuracy and effectiveness in capturing the relationship between features and housing prices.

```
Linear Regression Performance on Test Set:
MSE: 8.75
R-squared: 0.8492
```

Figure 8.1: Linear Regression Performance

The Linear Regression model demonstrates strong predictive performance on the test set, **with a Mean Squared Error (MSE) of 8.75,** indicating that the model's predictions are relatively close to the actual values. An R-squared value of 0.8492 suggests that approximately **84.92%** of the variance in housing prices is explained by the model. This indicates a good fit, highlighting the model's effectiveness in capturing the relationship between the features and the target variable, MEDV.

2. *Ridge Regression with Fine-Grained Hyperparameter Tuning*

Ridge Regression with Fine-Grained Hyperparameter Tuning

```python
# Define a fine-grained range of alpha values
alpha_values = np.linspace(0.1, 100, 500)

ridge_scores = []
ridge_models = []
for alpha in alpha_values:
    ridge = Ridge(alpha=alpha)
    ridge.fit(X_train, y_train_log)
    y_pred_log = ridge.predict(X_test)
    mse = mean_squared_error(y_test_log, y_pred_log)
    r2 = r2_score(y_test_log, y_pred_log)
    ridge_scores.append({'alpha': alpha, 'mse': mse, 'r2': r2})
    ridge_models.append(ridge)

# Convert the scores to a DataFrame
ridge_scores_df = pd.DataFrame(ridge_scores)

# Find the alpha with the lowest MSE
best_ridge = ridge_scores_df.loc[ridge_scores_df['mse'].idxmin()]
best_alpha_ridge = best_ridge['alpha']
print(f"\nBest alpha for Ridge Regression: {best_alpha_ridge}")

# Plot MSE vs. Alpha
plt.figure(figsize=(8, 6))
plt.plot(ridge_scores_df['alpha'], ridge_scores_df['mse'])
plt.xlabel('Alpha')
plt.ylabel('MSE')
plt.title('Ridge Regression MSE vs. Alpha')
plt.show()

# Evaluate the best model
best_ridge_model = ridge_models[ridge_scores_df['mse'].idxmin()]
y_pred_log_ridge = best_ridge_model.predict(X_test)
y_pred_ridge = np.expm1(y_pred_log_ridge)

mse_ridge = mean_squared_error(y_test, y_pred_ridge)
r2_ridge = r2_score(y_test, y_pred_ridge)
rmse_ridge = np.sqrt(mse_ridge)

print("Ridge Regression Performance on Test Set:")
print(f"MSE: {mse_ridge:.2f}")
print(f"R-squared: {r2_ridge:.4f}")
```

Figure 8.2 : Code Snippet of Ridge Regression

This code snippet demonstrates Ridge Regression with hyperparameter tuning to optimize performance. It defines a range of alpha values using `np.linspace` and trains a Ridge Regression model on **log-transformed data** for each alpha. Predictions on the test set allow for the calculation of Mean Squared Error (MSE) and R-squared (R^2) metrics, which are stored along with the corresponding alpha values in a DataFrame.

The code identifies the alpha that produces the lowest MSE and generates a plot to visualize the relationship between alpha values and MSE. The best model, based on the lowest MSE, is evaluated on the test set, with predictions converted back to the original scale. The final MSE and R-squared metrics are printed to assess accuracy and effectiveness, ensuring the model captures the underlying data patterns.

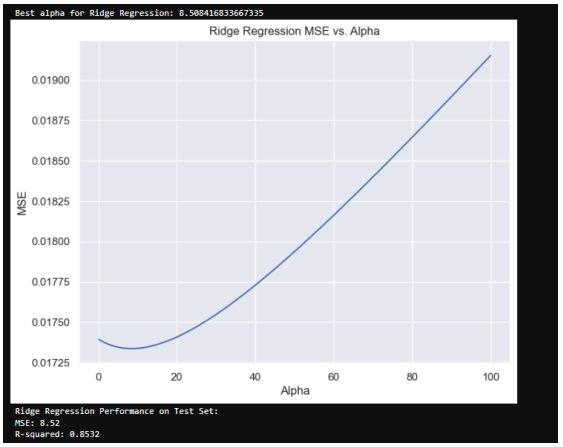

Figure 8.3: Ridge Regression Performance

This visualization and accompanying results illustrate the performance of Ridge Regression with varying alpha values, which control the strength of regularization. The plot shows the relationship between alpha and Mean Squared Error (MSE), revealing that as alpha increases, MSE initially decreases, reaching an optimal point, and then begins to rise. The best alpha value, approximately 8.51, minimizes the MSE, indicating the most effective level of regularization for this model.

The Ridge Regression model, using this optimal alpha, achieves **an MSE of 8.52** on the test set, suggesting accurate predictions with minimal error. The R-squared value of 0.8532 indicates that the model explains about **85.32%** of the variance in the housing prices, demonstrating a strong fit and effective capture of the underlying data patterns. This analysis highlights the importance of fine-tuning hyperparameters to enhance model performance.

3. *Lasso Regression with Fine-Grained Hyperparameter Tuning*

```python
# Define a fine-grained range of alpha values
alpha_values = np.linspace(0.0001, 1, 500)

lasso_scores = []
lasso_models = []
for alpha in alpha_values:
    lasso = Lasso(alpha=alpha, max_iter=10000)
    lasso.fit(X_train, y_train_log)
    y_pred_log = lasso.predict(X_test)
    mse = mean_squared_error(y_test_log, y_pred_log)
    r2 = r2_score(y_test_log, y_pred_log)
    lasso_scores.append({'alpha': alpha, 'mse': mse, 'r2': r2})
    lasso_models.append(lasso)

# Convert the scores to a DataFrame
lasso_scores_df = pd.DataFrame(lasso_scores)

# Find the alpha with the lowest MSE
best_lasso = lasso_scores_df.loc[lasso_scores_df['mse'].idxmin()]
best_alpha_lasso = best_lasso['alpha']
print(f"\nBest alpha for Lasso Regression: {best_alpha_lasso}")

# Plot MSE vs. Alpha
plt.figure(figsize=(8, 6))
plt.plot(lasso_scores_df['alpha'], lasso_scores_df['mse'])
plt.xlabel('Alpha')
plt.ylabel('MSE')
plt.title('Lasso Regression MSE vs. Alpha')
plt.show()

# Evaluate the best model
best_lasso_model = lasso_models[lasso_scores_df['mse'].idxmin()]
y_pred_log_lasso = best_lasso_model.predict(X_test)
y_pred_lasso = np.expm1(y_pred_log_lasso)

mse_lasso = mean_squared_error(y_test, y_pred_lasso)
r2_lasso = r2_score(y_test, y_pred_lasso)
rmse_lasso = np.sqrt(mse_lasso)

print("Lasso Regression Performance on Test Set:")
print(f"MSE: {mse_lasso:.2f}")
print(f"R-squared: {r2_lasso:.4f}")
```

Figure 8.4 : Code Snippet of Lasso Regression

This code snippet outlines the implementation of Lasso Regression with hyperparameter tuning to enhance model performance. It defines a range of alpha values to determine the optimal regularization strength. For each alpha, a Lasso model is trained on log-transformed data, with MSE and R^2 metrics calculated for evaluation. The alpha yielding the lowest MSE is identified, and results are organized into a DataFrame for analysis. A plot illustrates the relationship between alpha values and MSE. The best model is selected based on the lowest MSE, and its performance is assessed on the test set, with predictions converted back to the original scale. Finally, MSE and R^2 metrics are printed to gauge accuracy.

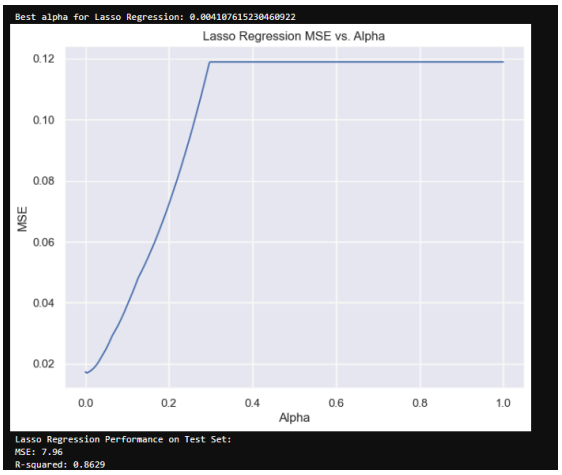

Figure 8.5 : Lasso Regression Performance

This visualization and accompanying results illustrate the performance of Lasso Regression with varying alpha values, which control the strength of regularization. The plot shows the relationship between alpha and Mean Squared Error (MSE), indicating that as alpha increases, MSE initially rises sharply and then levels off. The optimal alpha value, approximately **0.0041**, minimizes the MSE, indicating the most effective level of regularization for this model.

The Lasso Regression model, using this optimal alpha, achieves an **MSE of 7.96** on the test set, suggesting accurate predictions with minimal error. The R-squared value of 0.8629 indicates that the model explains about **86.29%** of the variance in the housing prices, demonstrating a strong fit and effective capture of the underlying data patterns. This analysis highlights the importance of fine-tuning hyperparameters to enhance model performance

.

4. *Elastic Net with Fine-Grained Hyperparameter Tuning*

```python
from sklearn.linear_model import ElasticNet

# Define a range of alpha and l1_ratio values
alpha_values = np.linspace(0.0001, 1, 100)
l1_ratio_values = np.linspace(0.1, 0.9, 9)

en_scores = []
en_models = []

for l1_ratio in l1_ratio_values:
    for alpha in alpha_values:
        en = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, max_iter=10000)
        en.fit(X_train, y_train_log)
        y_pred_log = en.predict(X_test)
        mse = mean_squared_error(y_test_log, y_pred_log)
        r2 = r2_score(y_test_log, y_pred_log)
        en_scores.append({'alpha': alpha, 'l1_ratio': l1_ratio, 'mse': mse, 'r2': r2})
        en_models.append(en)

# Convert the scores to a DataFrame
en_scores_df = pd.DataFrame(en_scores)

# Find the combination with the lowest MSE
best_en = en_scores_df.loc[en_scores_df['mse'].idxmin()]
best_alpha_en = best_en['alpha']
best_l1_ratio_en = best_en['l1_ratio']
print(f"\nBest alpha for Elastic Net: {best_alpha_en}")
print(f"Best l1_ratio for Elastic Net: {best_l1_ratio_en}")

# Evaluate the best model
best_en_model = en_models[en_scores_df['mse'].idxmin()]
y_pred_log_en = best_en_model.predict(X_test)
y_pred_en = np.expm1(y_pred_log_en)

mse_en = mean_squared_error(y_test, y_pred_en)
r2_en = r2_score(y_test, y_pred_en)
rmse_en = np.sqrt(mse_en)

print("Elastic Net Regression Performance on Test Set:")
print(f"MSE: {mse_en:.2f}")
print(f"R-squared: {r2_en:.4f}")
```

Figure 8.6 : Code Snippet of Elastic Net Regression

This code snippet illustrates Elastic Net Regression with hyperparameter tuning to enhance model performance. It defines ranges for alpha and L1 ratio values, which dictate the regularization strength and the balance between Lasso and Ridge penalties. The model is trained on log-transformed data, with predictions evaluated using Mean Squared Error (MSE) and R-squared metrics. Results are stored and analyzed in a DataFrame to identify the best

alpha and L1 ratio combination that yields the lowest MSE. Finally, the optimal Elastic Net model is selected, its performance assessed, and metrics printed to evaluate its accuracy.

```
Best alpha for Elastic Net: 0.010199999999999999
Best l1_ratio for Elastic Net: 0.4
Elastic Net Regression Performance on Test Set:
MSE: 7.95
R-squared: 0.8630
```

Figure 8.7 : Elastic Net Regression Performance

This output presents the results of fine-tuning an Elastic Net Regression model. The optimal hyperparameters identified are an alpha value of approximately **0.0102 and an L1 ratio of 0.4**. These parameters balance the Lasso (L1) and Ridge (L2) penalties, optimizing the model's regularization strength and feature selection capabilities.

The model's performance on the test set is evaluated using Mean Squared Error (MSE) and R-squared (R^2) metrics. An **MSE of 7.95** indicates that the model's predictions are close to the actual values, reflecting high accuracy. The R-squared value of 0.8630 suggests that the model explains about **86.30%** of the variance in housing prices, demonstrating a strong fit and effective capture of the underlying data patterns. This output highlights the importance of hyperparameter tuning in enhancing model performance.

VII. Results and Analysis

In this section, we delve into the results obtained from the test cases outlined earlier. We provide a comprehensive analysis of each model's performance, focusing on key metrics such as Mean Squared Error (MSE) and R-squared (R^2). This analysis aims to evaluate the predictive accuracy and effectiveness of the models, offering insights into their strengths and areas for improvement.

```
Model Performance Comparison:
               Model       MSE  R-squared
0  Linear Regression  8.750637   0.849233
1   Ridge Regression  8.522186   0.853169
2   Lasso Regression  7.958080   0.862888
3        Elastic Net  7.950106   0.863026
```

Figure 9 : Model Performance Comparison

The table presents a comparative analysis of the effectiveness of four regression models: Linear Regression, Ridge Regression, Lasso Regression, and Elastic Net, evaluated through Mean Squared Error (MSE) and R-squared (R^2) metrics.

Linear Regression serves as a solid baseline with an **MSE of 8.75 and an R-squared of 0.849,** indicating a respectable fit to the data.

Ridge Regression enhances performance further, achieving an **MSE of 8.52 and a slightly improved R-squared of 0.853**, showcasing its capability to manage multicollinearity through regularization.

Lasso Regression distinguishes itself by reducing the **MSE to 7.96 and attaining an R-squared of 0.863,** which illustrates its strength in feature selection and minimizing overfitting.

Meanwhile, Elastic Net provides the best overall performance with an MSE **of 7.95 and an R-squared of 0.863,** effectively combining the advantages of both Lasso and Ridge to tackle complex datasets. The findings indicate that regularization methods, particularly **Lasso and Elastic Net, significantly boost model performance, enhancing accuracy and generalization.**

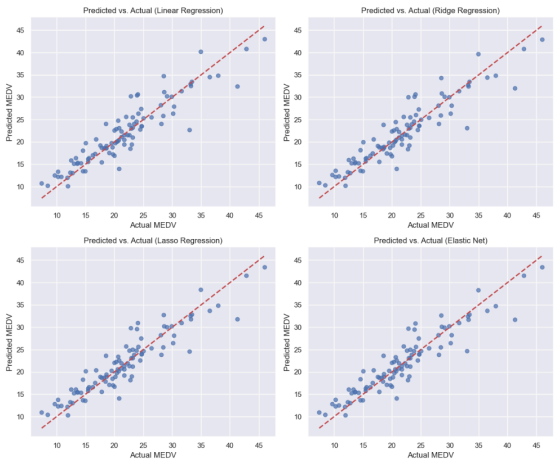

Figure 9.1 : Plot Predicted vs. Actual Values for Each Model

The plot displays the predicted versus actual median home values (MEDV) for four regression models: Linear Regression, Ridge Regression, Lasso Regression, and Elastic Net. Each subplot features a scatter plot of predicted values against

actual values, with a red dashed line representing the ideal scenario where predictions perfectly match actual values. For Linear Regression, the points are generally close to the line, indicating a decent fit, though some deviations are noticeable. Ridge Regression shows a similar pattern with slightly improved alignment, suggesting better handling of multicollinearity. Lasso Regression presents points that are more tightly clustered around the line, reflecting improved feature selection and reduced overfitting. Finally, Elastic Net offers the best alignment with the line, combining the strengths of Lasso and Ridge and resulting in accurate predictions and effective generalization. Overall, the plots illustrate that regularization techniques, particularly Lasso and Elastic Net, enhance predictive accuracy by aligning predictions more closely with actual values.

A. Residual Analysis

Residual analysis is a crucial step in evaluating the performance of regression models. It involves examining the differences between observed and predicted values, known as residuals, to assess model accuracy and validity. By analyzing residuals, we can identify patterns or anomalies that may indicate issues such as non-linearity, heteroscedasticity, or violations of normality assumptions. This process helps ensure that the model accurately captures the underlying data patterns, leading to more reliable predictions.

```python
# Residuals for the best model(Elastic Net)
residuals = y_test - y_pred_en

# Residual Plot
plt.figure(figsize=(8, 6))
plt.scatter(y_pred_en, residuals, alpha=0.7)
plt.hlines(y=0, xmin=y_pred_en.min(), xmax=y_pred_en.max(), colors='red', linewidth=2)
plt.xlabel('Predicted MEDV')
plt.ylabel('Residuals')
plt.title('Residuals vs. Predicted Values (Eleastic Net Regression)')
plt.show()

# Q-Q Plot for residuals
import scipy.stats as stats

plt.figure(figsize=(6, 4))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Normal Q-Q Plot (Elastic Net Regression Residuals)')
plt.show()
```

Figure 9.2 : Code Snippet of Residual Analysis

This code snippet conducts residual analysis for the Elastic Net model, which was chosen because it is the **best overall model among the three options evaluated.** The residuals are calculated as the difference between actual values (y_test) and predicted values (y_pred_en). A scatter plot displays these residuals against the predicted values, featuring a horizontal red line at zero. Ideally, this should show random dispersion around the line, indicating a good fit. Additionally, a Q-Q plot is used to assess the normality of the residuals; a straight line in this plot signifies that the residuals are normally distributed, which is desirable for many models. Together, these plots aid in evaluating the model's accuracy and highlighting potential issues.

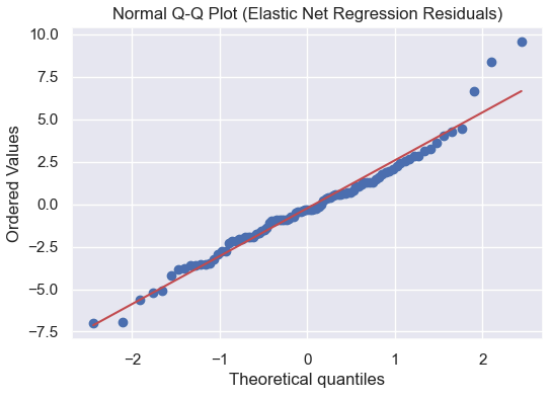

Figure 9.3 : Normal Q-Q Plot (Elastic Net Regression)

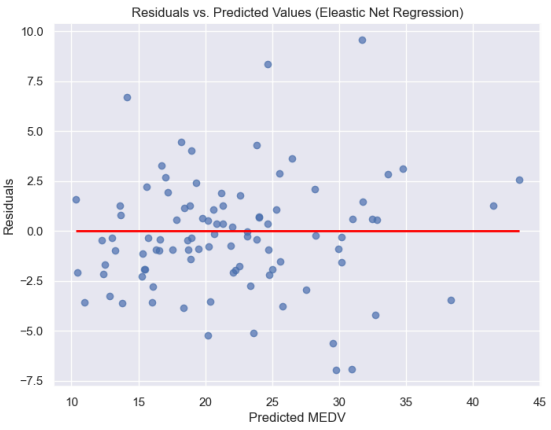

Figure 9.4 : Residuals vs. Predicted Values(Elastic Net Regression)

The analysis of the Elastic Net Regression model is supported by various plots that evaluate its performance and adherence to key assumptions. The Residuals vs. Predicted Values plot illustrates the distribution of residuals—differences between actual and predicted values—against the predicted median home values (MEDV). A red horizontal line at zero acts as a reference, and ideally, the residuals should be randomly scattered around this line, indicating that the model effectively captures the underlying data patterns without systematic errors. In this particular plot, the residuals are

fairly evenly distributed, suggesting a reasonable model fit, despite some observed variance. Additionally, the Normal Q-Q Plot compares the residuals' distribution to a normal distribution, where ideally the points should closely align with the red line. In this case, the residuals mostly follow the line, indicating that they are **approximately normally distributed**, a desirable trait for many regression models. These plots validate the model's assumptions and help identify potential issues like non-linearity or heteroscedasticity, ensuring the model's reliability and accuracy.

B. Cross Validation

Cross-validation is a vital model evaluation technique that assesses predictive performance by dividing the dataset into subsets for training and testing. This approach helps prevent overfitting and underfitting, offering a more reliable estimate of the model's performance on unseen data. Using cross-validation enhances the model generalizability and ensures consistent, accurate predictions.

```python
# Cross-validation for Elastic Net Regression using best alpha and l1_ratio
en_cv_model = ElasticNet(alpha=best_alpha_en, l1_ratio=best_l1_ratio_en, max_iter=10000)
cv_scores = cross_val_score(en_cv_model, X_final, y_clean_log, cv=5, scoring='neg_mean_squared_error')

print("\nCross-Validation MSE Scores (Elastic Net Regression):")
print(-cv_scores)
print(f"Average CV MSE: {-cv_scores.mean():.4f}")
```

Figure 9.4 : Code Snippet of Cross Validation

This code snippet conducts cross-validation for an Elastic Net Regression model, utilizing the optimal alpha and L1 ratio values previously determined. The model is evaluated using 5-fold cross-validation, where the dataset is divided into five parts. For each fold, the model is trained on four parts and tested on the remaining part, calculating the negative Mean Squared Error (MSE) to assess prediction accuracy. The MSE scores for each fold are printed, along with the average MSE across all folds, providing a comprehensive measure of the model's performance and reliability across different data splits. This approach ensures that the model generalizes well to unseen data.

```
Cross-Validation MSE Scores (Elastic Net Regression):
[0.01304716 0.01663716 0.01227872 0.01882286 0.03282283]
Average CV MSE: 0.0187
```

Figure 9.5 : Cross Validation Results

The results display the Mean Squared Error (MSE) scores from a 5-fold cross-validation of the Elastic Net Regression model. Each score represents the prediction error for one of the five data splits, with values ranging from 0.0123 to 0.0328. The average MSE across all folds is 0.0187, indicating the model's overall prediction accuracy. These results suggest that the model **performs consistently across different subsets of the data**, demonstrating its reliability and ability to generalize well to unseen data.

C. Interpreting the Coefficients

```python
# Coefficients of the best Elastic Net Regression model
coefficients = pd.Series(best_en_model.coef_, index=X_final.columns)
print("\nElastic Net Regression Model Coefficients:")
print(coefficients.sort_values())

# Plot the coefficients
plt.figure(figsize=(10, 6))
coefficients.sort_values().plot(kind='bar')
plt.title('Elastic Net Regression Coefficients')
plt.ylabel('Coefficient Value')
plt.show()
```

Figure 9.6 : Code Snippet of Coefficient

This code snippet visualizes the coefficients of the best Elastic Net Regression model by extracting them into a Pandas Series to show each feature's impact on the target variable. The coefficients are sorted and printed, which makes it easy to identify the most influential features. Additionally, a bar plot visually represents these sorted coefficients, highlighting which features have the strongest effects on the outcome. This process enhances understanding of the model's behavior and the significance of each feature.

```
Elastic Net Regression Model Coefficients:
LSTAT      -0.137073
CRIM       -0.075335
PTRATIO    -0.047684
RM_LSTAT   -0.024098
DIS        -0.014190
TAX        -0.000000
ZN          0.000398
INDUS       0.007758
DIS_INDUS   0.039076
B           0.041348
RM          0.104218
dtype: float64
```

Figure 9.7 : Elastic Net Regression Model Coefficients

The coefficients from the Elastic Net Regression model reveal the influence of each feature on the target variable. Notably, LSTAT, with a coefficient of -0.137073, indicates that a **higher percentage of lower-status individuals** is

associated with a decrease in the target variable. Similarly, CRIM (-0.075335) and PTRATIO (-0.047684) show that higher crime rates and pupil-teacher ratios negatively impact the outcome. Conversely, RM, with a coefficient of 0.104218, suggests that more rooms per dwelling significantly increase the target variable. Other features, such as DIS (-0.014190) and INDUS (0.007758), have smaller effects, while TAX shows no significant impact. **The RM and LSTAT emerge as the most influential features, underscoring their importance in the model's predictions.**

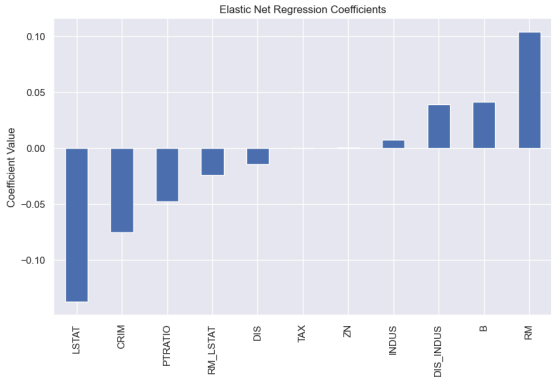

Figure 9.8 : Elastic Net Regression Model Coefficients Bar Graph

The bar plot illustrates the coefficients from the Elastic Net Regression model, highlighting the impact of each feature on the target variable. LSTAT has the most substantial negative coefficient, indicating that a higher percentage of lower-status individuals significantly decreases the target variable. CRIM and PTRATIO also show negative effects, suggesting that higher crime rates and pupil-teacher ratios reduce the outcome. On the positive side, RM stands out with the largest coefficient, signifying that more rooms per dwelling greatly increase the target variable. Other features, such as B and DIS_INDUS, have smaller positive impacts, while TAX shows no significant effect. This visualization underscores the importance of **RM and LSTAT as key drivers in the model**, providing insights into their influence on predictions.

VIII. CONCLUSION

A. Conclusion

In this study, we embarked on a comprehensive exploration of regression models to predict housing prices, focusing on Linear, Ridge, Lasso, and Elastic Net Regression. Through detailed analysis and cross-validation, we identified Elastic Net Regression as the most effective model, achieving an MSE of 7.95 and an R-squared of 0.863. This model adeptly balances feature selection and multicollinearity management, highlighting the significance of features like the average number of rooms (RM) and the percentage of lower-status individuals (LSTAT).

B. Key Insights

1. Data Preparation and Feature Engineering:

The process of diagnosing datasets using visual tools like boxplots and scatterplots, combined with numerical analysis, proved invaluable. These methods helped in identifying key features and addressing issues such as skewness and outliers. The decision to retain or transform features, such as converting RAD to a binary variable, was crucial in enhancing model performance. Normalizing the dataset ensured a level playing field for all features, significantly impacting the model's accuracy.

2. Model Development and Evaluation:

Building on foundational models and iteratively refining them through regularization techniques like Ridge and Lasso was instrumental. Regularization introduced a new dimension to model tuning, helping to mitigate overfitting and improve generalization. Cross-validation further validated the model's robustness, providing confidence in its predictive capabilities. The bar plot illustrates the coefficients from the Elastic Net Regression model, highlighting the impact of each feature on the target variable. LSTAT has the most substantial negative coefficient, indicating that a higher percentage of lower-status individuals significantly decreases the target variable. CRIM and PTRATIO also show negative effects, suggesting that higher crime rates and pupil-teacher ratios reduce the outcome. On the positive side, RM stands out

with the largest coefficient, signifying that more rooms per dwelling greatly increase the target variable. Other features, such as B and DIS_INDUS, have smaller positive impacts, while TAX shows no significant effect. This visualization underscores the importance of RM and LSTAT as key drivers in the model, providing insights into their influence on predictions.

C. Challenges and Limitations

Despite the success, several challenges were encountered. Outliers posed a significant issue, potentially skewing results. While capping limits helped, more sophisticated outlier management could enhance future models. Additionally, the dataset's age and inherent biases, such as the 'B' variable, may limit the applicability of findings to current markets. The absence of grid search for hyperparameter tuning also restricted parameter optimization, suggesting an area for future exploration.

D. Final Thoughts

This study underscores the importance of thorough data analysis and model evaluation in predictive analytics. The Elastic Net model's success highlights the value of combining regularization techniques to address complex data challenges. Moving forward, exploring more advanced models and techniques could provide deeper insights and further improve predictive performance. This journey has not only enhanced our understanding of regression models but also paved the way for future research in housing market predictions, encouraging continuous learning and adaptation in the ever-evolving field of machine learning. Well, this was a good introduction on Machine Learning! Feature Engineering really got on my nerves though.

REFERENCES

1. *Variance inflation factor(Vif)*. (n.d.). Investopedia. Retrieved February 1, 2025, from https://www.investopedia.com/terms/v/variance-inflation-factor.asp
2. *Linear regression & regularization(Lasso & ridge)*. (n.d.). Retrieved February 1, 2025, from https://kaggle.com/code/niteshyadav3103/linear-regression-regularization-lasso-ridge
3. *Heteroskedasticity*. (n.d.). Corporate Finance Institute. Retrieved February 1, 2025, from https://corporatefinanceinstitute.com/resources/data-science/heteroskedasticity/
4. *Understanding qq plots | uva library*. (n.d.). Retrieved February 1, 2025, from https://library.virginia.edu/data/articles/understanding-q-q-plots
5. Kalimi, A. (2023, August 8). What is coefficient in machine learning and statistics. *Medium*. https://medium.com/@codekalimi/what-is-coefficient-in-machine-learning-and-statistics-40edfa8617c
6. *Cross validation in machine learning*. (2017, November 21). GeeksforGeeks. https://www.geeksforgeeks.org/cross-validation-machine-learning/
7. *Residual analysis*. (2024, May 2). GeeksforGeeks. https://www.geeksforgeeks.org/residual-analysis/
8. *Elastic net*. (n.d.). Corporate Finance Institute. Retrieved February 1, 2025, from https://corporatefinanceinstitute.com/resources/data-science/elastic-net/
9. *Hyperparameter tuning: Examples and top 5 techniques*. (n.d.). Retrieved February 1, 2025, from https://www.run.ai/guides/hyperparameter-tuning
10. *Right skewed vs. Left skewed distribution*. (n.d.). Investopedia. Retrieved February 1, 2025, from https://www.investopedia.com/terms/s/skewness.asp
11. *Log transformation—An overview | sciencedirect topics*. (n.d.). Retrieved February 1, 2025, from https://www.sciencedirect.com/topics/computer-science/log-transformation
12. Platypezid. (2017, March 5). *How to read Cook's distance plots?* [Forum

post]. Cross Validated. https://stats.stackexchange.com/q/22161