Exploring Linear Regression Through Boston House Dataset

Advanced Intelligent Systems - Summative Lab Exercise #1

Alvarez, Aaron Jetro C.

I. INTRODUCTION

This paper centers on regression analysis as the statistical technique for building the model. This systematically explores each step of the modeling process, from data preprocessing, treatment, feature selection, and model evaluation. The models are examined using linear regression as the base model and its variants, specifically ridge regression, Lasso regression, and elastic net regression. The paper highlights how these techniques can be optimized to improve model accuracy and robustness. It also discusses the impact of each methodological choice on the model's performance upon experimentation, providing insight into regression-based modeling.

1.1 Dataset Overview

This paper uses the Boston House dataset to develop a predictive model. It is derived from information the U.S. Census Service collected concerning housing in Boston, MA. It is now a widely used data set as it is prebuilt into sklearn datasets, often used to learn machine learning. The following describes the data sets:

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

1.2 Problem Statement

The main objective of this activity is to build a predictive model to estimate the median house prices based on neighborhood statistics.

This objective can further be divided into specific parts, which are:

1. Understand the relationships and visualize data through exploratory data analysis
2. Handle abnormalities outliers and normalize/scale the dataset through data Preprocessing
3. Implement models that can predict the target value using preprocessed data and tuning the model for better performance.
4. Evaluate the models and compare the performance of different models.
5. Analyzing the impact of the coefficients and the significance of regularization, a technique used to prevent overfitting in predictive models, can narrow down the features that are most influential in predicting house prices.

1.3 Challenges

Before the experimentation, I outlined possible challenges I might encounter by building the model. These are a personal set of questions that I have posed and will try to answer indirectly throughout the experimentation process. I've made this section to reflect and systematically approach the said challenges.

1. In the presence of outliers, should it be removed or modified so that the model will penalize it less?
2. How do I incorporate two or more features that aren't target variables so that they can be used to predict the target value?
3. How should I identify the features that may not be significant for the target value and ensure that it can predict the target value correctly?
4. Can I modify the potential significant coefficients that may help predict values but are poorly modeled due to abnormalities?
5. What statistical approaches can I use if I encounter a skewed graph?

6. Since there are a lot of hyper-tuning parameters, how can I identify which combinations and models are the best?
7. If a model performs poorly, identify which step of the model building was poorly done and what approaches I can make to improve it.

II. METHODOLOGY

2.1 Exploratory Data Analysis

To gain intuition on how the data looks, we print out the data head to get what kind of values we are dealing with.

```
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.430483	6.575	65.2	1.627278	1	296.0	15.3	396.90	1.788421	24.0
1	0.02731	0.0	7.07	0	0.384582	6.421	78.9	1.786261	2	242.0	17.8	396.90	2.316488	21.6
2	0.02729	0.0	7.07	0	0.384582	7.185	61.1	1.786261	2	242.0	17.8	392.83	1.615420	34.7
3	0.03237	0.0	2.18	0	0.377066	6.998	45.8	1.954757	3	222.0	18.7	394.63	1.371181	33.4
4	0.06905	0.0	2.18	0	0.377066	7.147	54.2	1.954757	3	222.0	18.7	396.90	1.845300	36.2

Additionally, we could check for the data information so we could see how many values there per column are. This helps us see if there are missing values or there are unpaired data columns.

```
print(data.info())
print(data.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

We now check for further information of each column using data.describe()

```
              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    0.438587    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.072948    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.325700    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.370874    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.430483    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.484892    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.626473    8.780000

              AGE         DIS         RAD         TAX     PTRATIO           B
\
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    1.479648    9.549407  408.237154   18.455534  356.674032
std     28.148861    0.413390    8.707259  168.537116    2.164946   91.294864
min      2.900000    0.755934    1.000000  187.000000   12.600000    0.320000
25%     45.025000    1.131459    4.000000  279.000000   17.400000  375.377500
50%     77.500000    1.436855    5.000000  330.000000   19.050000  391.440000
75%     94.075000    1.822659   24.000000  666.000000   20.200000  396.225000
max    100.000000    2.574633   24.000000  711.000000   22.000000  396.900000

            LSTAT        MEDV
count  506.000000  506.000000
mean     2.475610   22.532806
std      0.539033    9.197104
min      1.004302    5.000000
25%      2.073162   17.025000
50%      2.514464   21.200000
75%      2.887869   25.000000
max      3.662792   50.000000
```

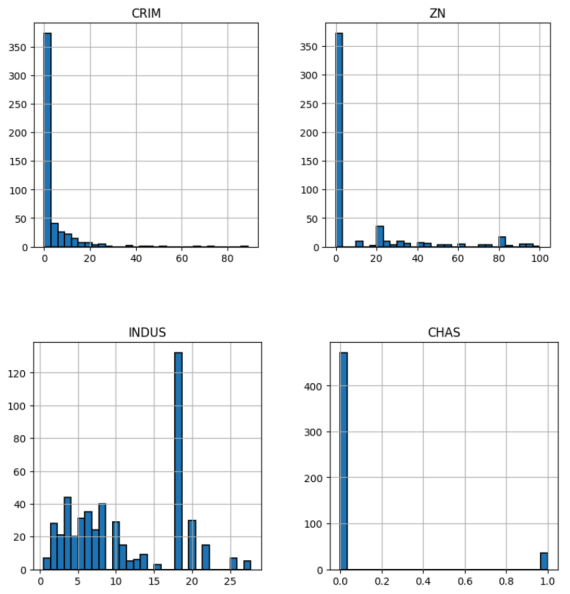
To further solidify if there are no missing values, we check if there are null values in the data set.

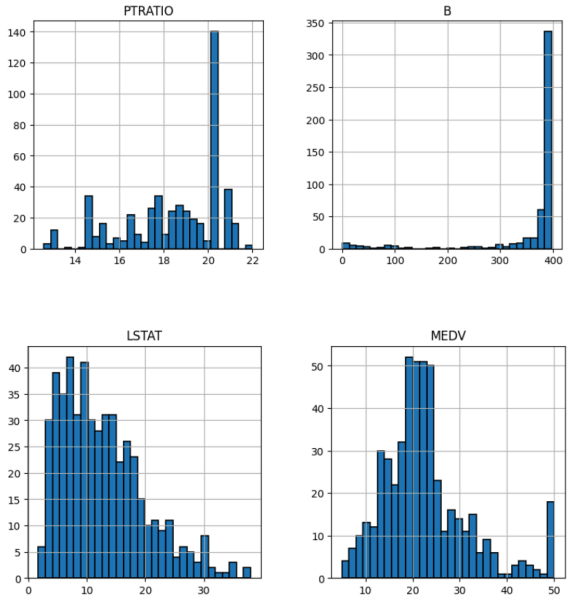
```
print(df.isnull().sum())
```

```
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
```

Now that the data information has been verified, we can use visual aids to help us identify correlations and possible outliers for the features.

My first task was to gain a comprehensive understanding of the dataset distribution. To achieve this, I employed Histograms, a powerful tool that allows us to discern between normally distributed and skewed datasets.

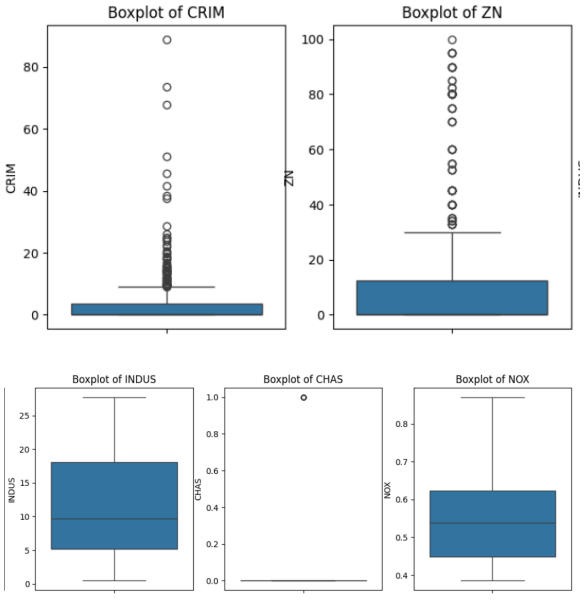


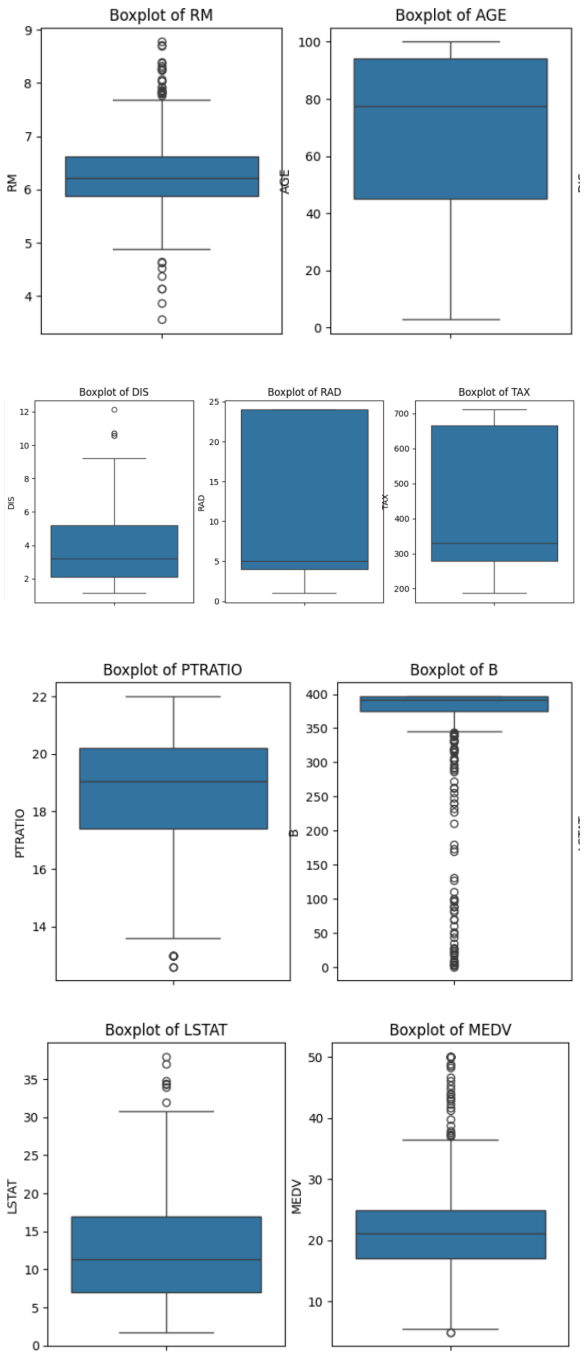


Histogram of all Data

I can already see that most features don't have a normal distribution (Gaussian bell shape) except for RM and MEDV. LSTAT, DIS is positively skewed, while AGE is negatively skewed. The rest doesn't follow a distribution due to fluctuating data points and extreme values (which leads me to suspect that there are outliers in the data set)

To find outliers in the data set, I used box plots to visualize the quartiles and see if the data sets were outside the mean quartile range.

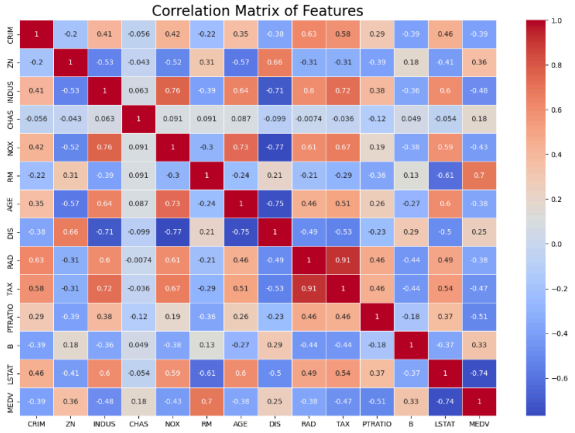




Box plot of All data

The boxplot was indeed able to show information that could not be seen in the histogram, here we could see that 9 features have outliers, namely CRIM, ZN, CHAS, RM, DIS, PTRATIO, B, LSTAT and MEDV. We could also see that the median line for some of the features are not centered in the interquartile range (IQR) which solidifies the skewness of the data seen in the histograms earlier.

Now we check for the correlation of features using the heatmap feature of seaborn. This will help in identifying potential relevant features that can be used for the model to predict the target variable MEDV.



Coefficient Matrix of Dataset

We can see four potential variables: RM, LSTAT, PTRATIO, and INDUS, which have correlations 0.7, -0.73, 0.51, and -0.48 correlation scores to MEDV, respectively.

Looking further at the heatmap, we can also see signs of multicollinearity because other features correlate to other features, such as TAX to RAD, with a 0.91 correlation. NOX to INDUS with 0.76 correlation, DIS and NOX with -0.77 correlation but have low correlation to MEDV.

Scatterplots were also used to see the relationship of each predictor variable with the target variable MEDV. (Which is shown in the notebook). For this paper, the correlation matrix is enough to see which variables directly correlate to MEDV.

2.2 Data Preprocessing

For data preprocessing, many choices and combinations of preprocessing were available due to the nature of the features identified during the exploratory data analysis. This process must address:

- Outliers
- Skewed datasets
- Multicollinearity
- Feature Selection
- Feature Engineering

We already saw that there were no missing values, but there were outliers. So, we proceed to data splitting before handling the outliers.

2.2.1 Data splitting

The data splitting process is done by splitting the data into training and test sets. This allows us to assess the model's performance with unseen data (test sets). I also split the dataset before treating outliers and performing certain transformations. I want to avoid data leakage, particularly preprocessing, where information is leaked from future data (which is the test data in this case). A test size of 0.2 with a random state of 42 was used in the train_test split method.

This concept is based on a book written by Max Kuhn and Kjell Johnson, titled, Feature Engineering and Selection: A Practical Approach for Predictive Models, simply stating that indirect data leakage can also happen when data preparation techniques are applied to the entire data set. It indicates that when a dataset is, for example, scaled by the global and maximum values and split into train and test datasets, the test data set already has more information about the global distribution of the variable than it should. This often leads to overly optimistic performance metrics and poor generalization of new data.

2.2.2 Feature Selection

According to an article written by Microsoft about data mining. Feature selection helps solve two problems: having too much or too little high-value data. Additionally, we want to avoid information leakage when we treat the data by giving it information about other variables or predictors that it shouldn't know.

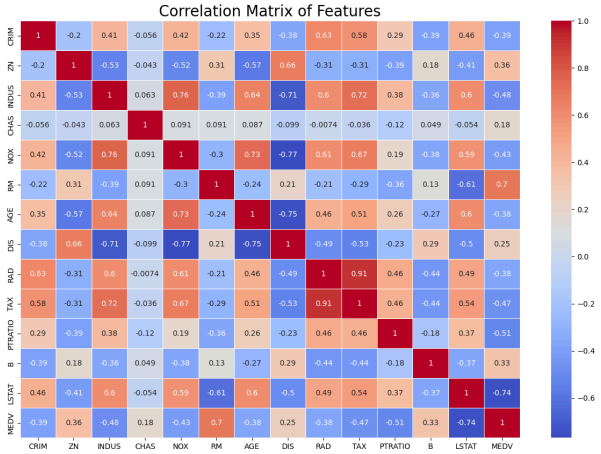
Since we don't want to over-clean the features by removing outliers and treating the skewness of graphs on all features, I decided to do feature selection first to narrow down which features should be treated (if necessary). I used features with only +- .5 correlation with 'MEDV.' This was done by assessing the correlation value taken from the earlier heatmap, which was sorted through a bit of sorting, as seen in the table below.

Top 5 Features with Positive Correlation to MEDV		
1	RM	0.695360
2	ZN	0.360445
3	B	0.333461
4	DIS	0.249929
5	CHAS	0.175260

Top 5 Features with Negative Correlation to MEDV		
1	LSTAT	-0.737663
2	PTRATIO	-0.507787
3	INDUS	-0.483725
4	TAX	-0.468536
5	NOX	-0.427321

The features that were selected were RM [0.70], PTRATIO[-0.51] and LSTAT[0-.74].

Another reason to do feature selection was because there were occurrences of multicollinearity as seen in the coefficient matrix.



Correlation matrix of features

This could cause unstable coefficients, inflated standard errors and overfitting. To solidify the information about multicollinearity I decided to use Variance Inflation Factor (VIF) to compute the multicollinearity of the predictors with other predictors.

```
Variance Inflation Factor (VIF):
    Feature   VIF
0      CRIM  1.71
1        ZN  2.47
2     INDUS  3.88
3      CHAS  1.10
4       NOX  4.47
5        RM  1.95
6       AGE  2.99
7       DIS  4.17
8       RAD  7.66
9       TAX  8.94
10  PTRATIO  1.85
11        B  1.33
12    LSTAT  2.82
```

TAX and RAD are seen to have high correlation with other predictors so if I decide to use one of them, I will have to omit one, or feature engineer it so that it can be used in some sort of way.

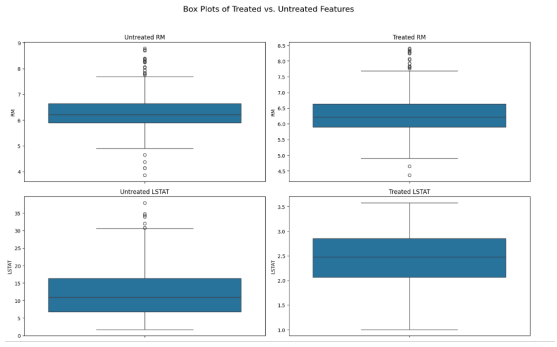
2..2.3 Outlier Treatment

During this stage, I had two possible options to address the outliers. It was to either:

- Remove the outliers using the IQR method where we drop data points outside of the range of the IQR.
- Remove the outliers by identifying their individual z scores based on the standard deviation of the data.

Z Score removal assumes that data is normally distributed; it calculates how many standard deviations a data point is from the mean. The IQE method calculates the range between the first and third quartiles but does not assume a particular distribution.

For this project, I decided to use z scores-based removal since, for this case, it's gentler in removing outliers, which can be crucial considering that I only have a small sample. Additionally, Since the data sample is small, IQR may remove data points that are not outliers. This means outlier removal will only be used for selected features with normal distribution.



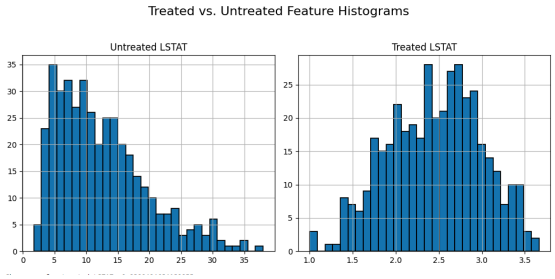
Box plot of Treated Data

I removed the outliers from the training test and the test set separately, and I only removed outliers from RM and LSAT since PTRATIO is not normally distributed.

2.2.4 Data Transformation

It was seen earlier that some of the graphs were either positively skewed or negatively skewed. From the features that I selected (RM, LSTAT, and PTRATIO), LSTAT has a positively skewed graph, so I used log transformation to treat the skewness of the data, which seemed effective, as seen in the results.

```
Skewness of untreated LSTAT: 0.9302494684182955
Skewness of treated LSTAT: -0.15118233318832475
```



Histogram of Treated Data

2.2.5 Normalization

Upon viewing the data earlier, using data. head(). We can see that the features have varying ranges (e.g., 0-100, then others 0-1). Additionally, some of the data has been treated and modified, such as when outliers were removed. This may cause an imbalance among the features where other features may dominate other features. Therefore, scaling the data should be necessary.

```
# Fit scaler on training data and transform both training
X_train_scaled = scaler.fit_transform(X_train_scaled)
X_test_scaled = scaler.transform(X_test_scaled)
```

For this data scaling was done through Standard Scaler which standardizes features by removing the mean and scaling to unit variance. This results in a distribution with a mean of 0 and standard deviation of 1.

Here we can see the effect of the transformation process:

`XOld #unscaled`

	RM	LSTAT	PTRATIO
477	5.304	24.91	20.2
15	5.834	8.47	21.0
332	6.031	7.83	16.9
423	6.103	23.29	20.2
19	5.727	11.28	21.0
...	...	...	...
106	5.836	18.66	20.9
270	5.856	13.00	18.6
348	6.635	5.99	17.0
435	6.629	23.27	20.2
102	6.405	10.63	20.9

`X_train #scaled`

	RM	LSTAT	PTRATIO
477	-1.547654	1.484774	0.849727
15	-0.736124	-0.394984	1.210368
332	-0.434480	-0.525669	-0.637918
423	-0.324235	1.364193	0.849727
19	-0.899961	0.090304	1.210368
...	...	...	...
106	-0.733062	0.969231	1.165288
270	-0.702438	0.335121	0.128444
348	0.490357	-0.962083	-0.592838
435	0.481170	1.362654	0.849727
102	0.138184	-0.011265	1.165288

2.3 Model Implementation and Evaluation

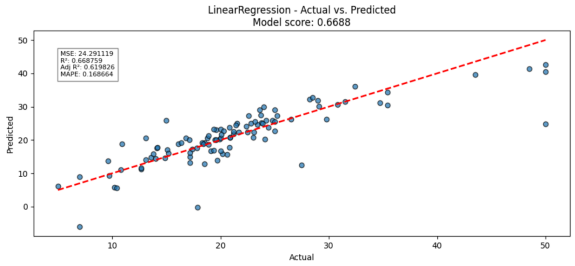
For this section, the model results are not yet final; I consider this the preliminary stage of building the models. I've included how I implemented each

model, which has two variations of each type. I implemented a model that uses pre-processed and unprocessed data to narrow down the potential factors that affect the model's performance. These comparisons were evaluated using R-squared, MSE, and MAPE as indicators of how well the model performs. Detailed process hyperparameter tuning and alternative methodologies were discussed in the Experimentation section, and the final results were discussed in the Results and Analysis section.

2.3.1 Baseline Model

For the baseline model, I wanted to know how well the model would perform if the data weren't treated and only scaled, including all of the data, regardless of the correlation. I chose this approach because I wanted to know if my preprocessing methodology improved the model.

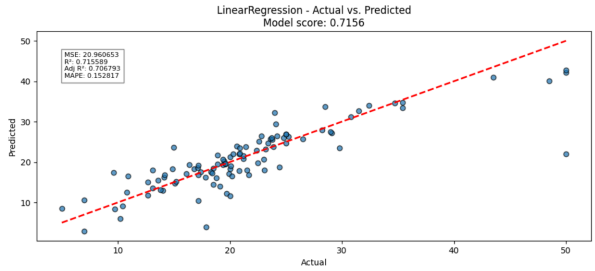
This is a linear regression model that uses all 13 features to predict the target value MEDV.



Scatterplot of Base model

Here, we observe the model score, which is 0.6686. This means that the model explains approximately 66.86% of the variance in the target variable. The model does not explain the remaining needs to be 14% of the variance. This indicates room for improvement in the model, as it can become more accurate with further optimization and data preprocessing.

We will now establish a model incorporating the data preprocessing steps performed earlier, which should enhance the accuracy and performance of the linear regression model.



Scatterplot of Base mode with Selected Features

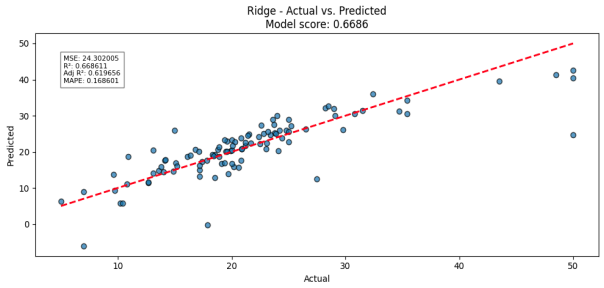
From this, we get an R-squared value of 0.7156 and a use of 20.96, which is better than the previous model.

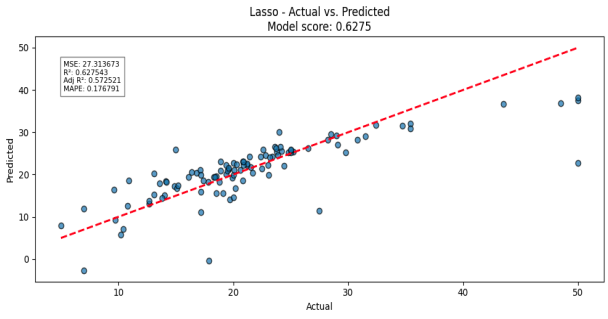
2.3.2 Advanced Model

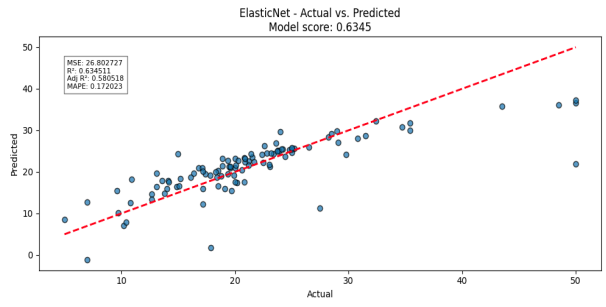
For the advanced models, the task objective specified Ridge Regression, Lasso Regression, and Elastic Net Regression. Upon research, the following models are variants of linear regression with regularization parameters to help address the sensitivity of linear regression to outliers, multicollinearity, and non-polynomial features. For this, I decided to make the following models using all of the features and then using the selected features to gain insight into how regularization works.

Using all Features

To see a fair comparison between the advanced models, I decided to use **alpha value 0.5** on all models, additionally I used **0.5 l1 value** on Elastic Net Model.



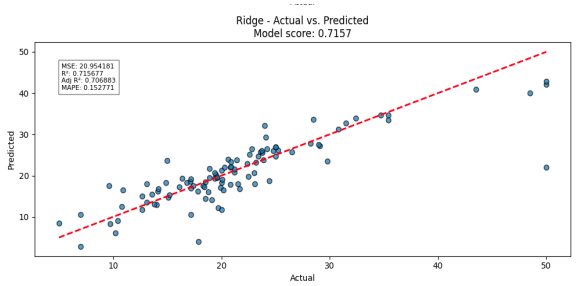


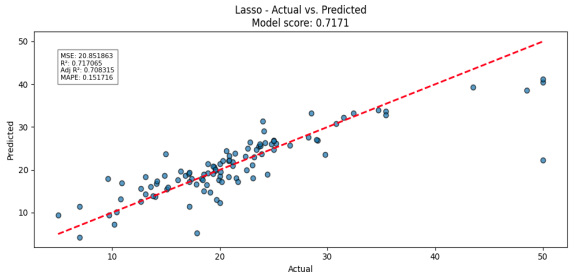


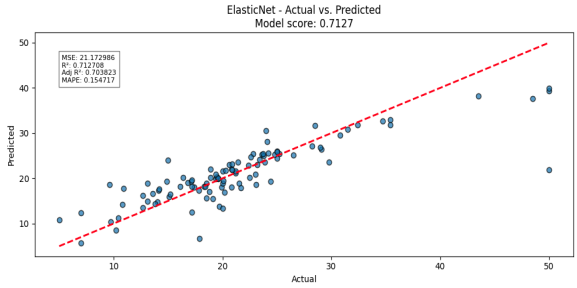
The only visual difference that could be observed for the models was the distance of some of the outliers. We're closer to the lines per model. The performance was slightly lower than the base model, but I suspect this was because the alpha value was too big or too small to impact the model.

Using Selected Features

This time I've used the same models, with the same parameters, except I only used selected features which were RM, PTRATIO and LSTAT.







If we compare it to the baseline model that uses only selected features, the models have a slightly lower performance. This is the same case with the baseline model vs advanced models using all features.

III. Experiments

Given that the 'pre-processed' vs. 'unprocessed' models almost gave out the same result, just having a higher R2 score solidifies the need for further tuning of the parameters of the advanced model since they have nearly the same sensitivity and predictive power regardless of the data.

In this section, I'll only show my experimentation and thought process for each experiment. I'll only highlight the changes in the data and explain what is happening. In-depth analysis and interpretation will be discussed more thoroughly in the Results and Analysis section.

3.1 Hyperparameter Tuning Models

Manual Tuning

From the word itself, the first type of tuning done was manual tuning. It involves exhaustively trying different alpha values and combinations of parameters.

Since linear regression models are sensitive to outliers and multicollinearity, I used pre-processed data as the model's features while tuning. Before the tuning process, here is the tabulated result of the models as the baseline of the performance:

Model	MSE	R^2	Adj R^2	MAPE
Linear	20.9607	0.7156	0.7068	0.1528
Ridge	20.9542	0.7157	0.7069	0.1528
Lasso	20.8519	0.7171	0.7083	0.1517
ElasticNet	21.1730	0.7127	0.7038	0.1547

Performance Metrics of Base Models with alpha 0.5

For tuning the models I had 3 basic categories, which were: **high**, **low** and **middle**. This allowed me to see the drastic effect of the hyperparameters given the categories.

Tuning the Ridge Regression Model

Based on the scikit-learn documentation, Ridge Regression solves a regression problem by minimizing the linear least squares loss function while incorporating L2 regularization to address issues like multicollinearity and overfitting. L2 regularization works by adding a penalty proportional to the square of the magnitude of the coefficients to the loss function. This encourages smaller coefficients, effectively shrinking them towards zero, but unlike L1 regularization, it does not force them to be precisely zero.

Tuning the Lasso Regression Model

Lasso regression is another regularization technique similar to Ridge but with L1 regularization. L1 regularization works by adding a penalty proportional to the sum of the absolute values of the coefficients to the loss function. This encourages

sparsity in the coefficient values, effectively shrinking some coefficients precisely to zero. This feature selection property helps reduce model complexity by eliminating less significant predictors. A higher alpha value increases the penalty on significant coefficients, so we will experiment with the alpha coefficients and analyze the trade-off between model fit and feature selection.

Tuning the Elastic Net Regression Model

Elastic regression is a mix of both Ridge and Lasso regression models. Where both penalty variables come from lasso and ridge regularizations. In elastic regression, two hyperparameter keys are alpha: alpha controls the strength of the amount of regularization applied. The L1 ratio controls the mix of L1 and L2 regularization where if the value is 1, the penalty is purely lasso, and if 0, the penalty is purely ridge.

Hyperparameter Tests Tabulated Results:

Model	MSE	R^2	Adj R^2	MAPE
Linear	20.9607	0.7156	0.7068	0.1528
Ridge	20.6614	0.7196	0.7110	0.1498
Lasso	75.3435	-0.0223	-0.0539	0.3762
ElasticNet	75.3435	-0.0223	-0.0539	0.3762

Alpha Value: High = 50

Model	MSE	R^2	Adj R^2	MAPE
Linear	20.9607	0.7156	0.7068	0.1528
Ridge	20.6614	0.7196	0.7110	0.1498
Lasso	75.3435	-0.0223	-0.0539	0.3762
ElasticNet	75.3435	-0.0223	-0.0539	0.3762

Alpha Value: Mid = 5

Model	MSE	R^2	Adj R^2	MAPE
Linear	20.9607	0.7156	0.7068	0.1528
Ridge	20.9605	0.7156	0.7068	0.1528
Lasso	20.9513	0.7157	0.7069	0.1528
ElasticNet	20.9310	0.7160	0.7072	0.1526

Alpha Value: Low = 0.01

GridSearchCV Tuning:

After observing the effects of the hyperparameters on model performance, I used GridSearchCV to automate the process of finding the best combination of hyperparameters. Instead of manually tuning each parameter on an excellent performance, this method systematically searches through a predefined set of hyperparameter values to identify the optimized configuration that yields the best performance among the predefined parameters through an exhaustive search.

Model and parameter grids were predefined before the actual search.

```
# Define models and their parameter grids
models = {
    'LinearRegression': LinearRegression(),
    'Ridge': Ridge(),
    'Lasso': Lasso(),
    'ElasticNet': ElasticNet()
}

param_grids = {
    'LinearRegression': {},
    'Ridge': {'alpha': [0.1, 0.5, 1.0, 10.0, 50.0]},
    'Lasso': {'alpha': [0.01, 0.1, 0.5, 1.0, 10.0]},
    'ElasticNet': {'alpha': [0.01, 0.1, 0.5, 1.0],
                   'l1_ratio': [0.1, 0.5, 0.7, 1.0]}
}
```

GridSearchCV was then setup with the following parameters:

```
grid_search = GridSearchCV(model,
                           param_grids[name],
                           cv=5,
                           scoring='neg_mean_squared_error',
                           n_jobs=-1)
'
```

This GridSearchCV uses negative mean squared error (MSE) to evaluate model performance. Since MSE is a loss function, we use the negative value of it to obtain the highest scores. A 5-fold cross-validation was used, which splits the training data into five subsets and is trained and validated five times.

After performing GridSearchCV for the Ridge, Lasso, and Elastic Net models, we obtain the following scores and parameters:

Model	Best Parameters	Best Score
Linear	(None)	22.7244
Ridge	'alpha': 10.0	22.6832
Lasso	'alpha': 0.01	22.7246
ElasticNet	'alpha': 0.01, 'l1_ratio': 0.1	22.7067

Using the best parameters found using Gridsearch, we obtain the following models:

Model	MSE	R^2	Adj R^2	MAPE
Linear (base model)	20.9607	0.7156	0.7068	0.1528
Ridge	20.8471	0.7171	0.7084	0.1519
Lasso	20.9513	0.7157	0.7069	0.1528
ElasticNet	20.9154	0.7162	0.7074	0.1525

There is a slight increase in the performance of the models compared to the base model with ridge having the highest R-squared score and lowest MSE.

RandomizedSearchCV Tuning

RandomizedSearchCV searches over specified parameter distribution by randomly sampling several parameter combinations. Compared to the exhaustive search of GridSearchCV, this samples a specified number of combinations. The problem is that it may miss out on optimal combinations if the number of samples is too low.

```
# Define models and their parameter distributions
models = {
    'LinearRegression': LinearRegression(),
    'Ridge': Ridge(),
    'Lasso': Lasso(),
    'ElasticNet': ElasticNet()
}

param_distributions = {
    'LinearRegression': {},
    'Ridge': {'alpha': uniform(0.1, 50)},
    'Lasso': {'alpha': uniform(0.01, 10)},
    'ElasticNet': {'alpha': uniform(0.01, 10),
                   'l1_ratio': uniform(0.1, 0.9)}
}
```

Compared to Gridsearch, we will specify a range of numbers that can be picked in the process.

```
random_search = RandomizedSearchCV(model, param_distributions[name],
                                   n_iter=10,
                                   cv=5,
                                   scoring='neg_mean_squared_error',
                                   n_jobs=-1,
                                   random_state=42)
```

Model	Best Parameters	Best Score
Linear	(None)	22.7244
Ridge	'alpha': 7.901	22.6872
Lasso	'alpha': 0.5908	23.2840
ElasticNet	'alpha': 0.2158, 'l1_ratio': 0.9729	22.7930

Using the best parameters found using RandomSearch, we obtain the following models:

Model	MSE	R^2	Adj R^2	MAPE
Linear	20.9607	0.7156	0.7068	0.1528
Ridge	20.8683	0.7168	0.7081	0.1521
Lasso	20.9124	0.7162	0.7075	0.1521
ElasticNet	20.8063	0.7177	0.7090	0.1518

Overall, RandomizedSearchCV gave almost similar results in comparison to GridSearchCV. But it was able to give a slightly better performance for Elastic Net Regression.

3.2 Preprocessing Methodologies

Now that the models have been tuned and performed at an acceptable level. I tried experimenting with my preprocessing methods, removing outliers, treating skewness, feature selection, and feature engineering.

Removing Outliers: Necessary or Unnecessary

Based on a research thread posted in ResearchGate about removing outliers (Wilhelm,J. 2015), removing outliers may be unnecessary or harmful to the model's performance. Especially if the samples are small and outliers may not be outlying, the rest of the values show together more tightly. However, some outliers may harm the regression line if it has considerable leverage. To test this, I made a model that does not have outlier removal and other uses outlier removal, which results in this:

Model	MSE	R^2	Adj R^2	MAPE
Linear	22.0016	0.7000	0.6908	0.1541
Ridge	21.9634	0.7005	0.6913	0.1534
Lasso	21.9903	0.7001	0.6910	0.1540
ElasticNet	21.9146	0.7012	0.6920	0.1526

For comparison here is the performance of the model with outlier treatment

Model	MSE	R^2	Adj R^2	MAPE
Linear	20.9607	0.7156	0.7068	0.1528
Ridge	20.8471	0.7171	0.7084	0.1519
Lasso	20.9513	0.7157	0.7069	0.1528
ElasticNet	20.9154	0.7162	0.7074	0.1525

Note that both tests used the same parameter taken from the GridSearchCV and that skewness of LSTAT was treated for both.

Removing Outliers: Before or After Data Split

As mentioned in the methodology section, data splitting was done before treatment to avoid data leakage. However, to further show the significant impact of the data leak, I decided to show a scenario where data is treated before data splitting. The same treatment and features were used for this experiment, except it was done before splitting the dataset into training and test sets. GridsearchCV was used to get the best parameters for all the models.

Model	MSE	R^2	Adj R^2	MAPE
Linear	10.9282	0.8329	0.8277	0.1404
Ridge	10.9181	0.8331	0.8279	0.1404
Lasso	10.9321	0.8329	0.8277	0.1404
ElasticNet	10.8946	0.8334	0.8283	0.1404

The MSE and R-Squared values are compared to the data treated after splitting. These values have a more optimistic performance, especially for a regression model sensitive to multicollinearity and outliers. The significant increase in performance is a clear indication of data leakage. The model's performance before splitting is artificially inflated because it can access information from the entire dataset.

Addressing Multicollinearity

The coefficient matrix showed signs of multicollinearity in the Methodology section. According to Statistics by Jim Frost, multicollinearity causes two problems:

1) Coefficient estimates can swing wildly based on other independent variables present in the model. These become very sensitive to small changes in the model.
2) Multicollinearity reduces the precision of the estimated coefficients.

This can lead to models that are hard to interpret, as noise from other coefficients can indirectly affect the prediction of the target variable.

To explore this further, I conducted experiments with three different combinations of predictors to see how multicollinearity impacts model performance. The combinations are:

1. Using all predictors
2. Using predictors with >5 VIF and directly correlated predictors to target variables.
3. Using only directly correlated predictors (>= 0.50 correlation) to the target variable.

The results of the Elastic Net models with alpha = 0.5 and l1 ratio = 0.5 tuning and no additional preprocessing are as follows:

Setup	MSE	R^2	Adj R^2	MAPE
1	24.3744	0.6676	0.6185	0.1683
2	25.9640	0.6459	0.6275	0.1640
3	26.6992	0.6359	0.6248	0.1747

At first glance, setup 1 appears to perform better, with a higher R^2 and lower MSE. However, this "better" performance is misleading due to the high VIF values, indicating multicollinearity. The high R^2 is inflated, and the model's coefficients are less reliable, as seen in the VIF values:

Feature	VIF	Feature	VIF
CRIM	1.71	CHAS	1.10
ZN	2.47	NOX	4.47
INDUS	3.88	AGE	1.95
RM	1.65	DIS	2.99
LSTAT	1.67	RAD	7.66
PTRATIO	1.18	TAX	8.94
B	1.33		

In contrast, here is setup 2, that removes other predictors but still maintains the 2 predictors with VIF > 5 which are TAX and RAD.

Feature	VIF
RM	1.75
LSTAT	2..07
PTRATIO	1.40
RAD	6.48
TAX	6.50

Notice how the VIF values of RAD and TAX went down when there were no other predictors used other than LSTAT, RM and PTRATIO. It went down indicating that RAD and TAX were actually influencing other variables. Now we can see the VIF for selected features only here in setup 3:

Feature	VIF
RM	1.65
LSTAT	1.67
PTRATIO	1.18

The VIF are almost close to 1 meaning they have little impact on other predictors other than the target variable itself.

Creating Correlated Variables

Since we've removed a lot of predictors, I wanted to see if I can use the omitted predictors to create new variables that can be used for the target variable without causing heavy multicollinearity. My approach was to pick three pairs of highly correlated variables and try to use them in the model.

For this case, I used the TAX and RAD pair with a 0.91 correlation, NOX and INDUS with a 0.77 correlation, and DIS and AGE with a -0.76 correlation.

Model	MSE	R^2	Adj R^2	MAPE
Linear	20.7988	0.7178	0.6998	0.1524
Ridge	20.6830	0.7194	0.7014	0.1514
Lasso	20.5760	0.7208	0.7030	0.1510
ElasticNet	20.5760	0.7208	0.7030	0.1510

With preprocessing and treatment applied, the model performs slightly better compared to the models that use only RM, LSTAT, PTRATIO. Here, we can also see that the VIF values for each of the added variables are low:

Feature	VIF
RM	2.00
LSTAT	2.64
PTRATIO	1.45
TAX_RAD	1.57
NOX_IND	1.21
DIS_AGE	1.22

This means, we were able to integrate 6 more predictors while avoiding multicollinearity. However the performance of the models does not have much of a significant improvement. Since none of these variables are not significantly correlated to the target variable MEDV itself.

IV. Summary of Results and Analysis

4.1 Initial Performance Metrics

In this section, we will be evaluating the model performance based on performance metrics and how each experiment had an impact on the performance.

Base Model

The linear regression model was used as the base model. where all 13 predictors were used with no data preprocessing and only feature scaling.

Model	MSE	R^2	Adj R^2	MAPE
Linear	24.2911	0.6688	0.6198	0.1687

The Mean Squared Error is generally high, reflecting that the predicted and actual values are very different. Likewise, an R-squared value of about 67% would indicate that the model could explain around 67% of the variance of the MEDV. In any case, however, improvement in the model for higher explanation power would be required. Even though the Adjusted R-squared, considering the number of predictors, is little improved compared to R-squared, enlightening the fact that further addition of predictors does not necessarily improve model performance. Additionally, a Mean Absolute Percentage Error (MAPE) of 16.87% depicts, on average, the extent to which the model predictions are away from the actual values; thus, it gives the model's level of accuracy for its predictions.

Advanced models

While the linear regression model serves as a good base model, results from the experiments show limitations like not having regularization, handling multicollinearity, handling noise, and reduced predictive performance. With that in mind, I decided to see how well its advanced models would handle if we were to use the same set of features that we used for the linear regression model:

Model	MSE	R^2	Adj R^2	MAPE
Ridge	24.3020	0.6686	0.6197	0.1686
Lasso	27.3137	0.6275	0.5725	0.1768
Elastic Net	26.8025	0.6345	0.5805	0.1720

While the ridge model performs close to the linear model, the other two models,lasso and elastic net, seem to have a lower value. It might be counter intuitive to think that a more advanced model performs more poorly than its base model, but this actually shows that the models capture more information than the linear regression model.

Ridge Regression

In the earlier sections, ridge regression is similar to linear regression, especially when the alpha value is small. This is because ridge regression adds L2 regularization to control coefficient size with smaller coefficients. If this parameter is small or the predictors are not shrink enough (penalized), it will show almost the same value as linear regression models.

Lasso Regression

Lasso adds a penalty proportional to the absolute values of coefficients, which can shrink some to zero, performing feature selection. We already established in the earlier experiments that the Boston dataset contains multicollinearity; this can lead to over-regularization, making the model too simplistic and causing underfitting. As a result, Lasso may need help generalizing the data and identifying clear patterns in it.

Additionally, the lower value of the adjusted r-squared in comparison to the r-squared reflects the model's ability to generalize and capture essential patterns in this current setting; the model cannot recognize the pattern due to solid regularization.

Elastic Net Regression

Elastic net combines l1 and l2 penalties. This approach. It balances between feature selection and coefficient shrinkage through its l1 ratio parameter. We can see that Elastic Net performs better than Lasso but no better than Ridge. This shows the trade-off capacity between l1 and l2 regularization to avoid extreme coefficient shrinkage. However, this also indicates that the model is not tuned correctly, showing signs of overregularization.

A smaller Adjusted R-squared compared to R-squared in Elastic Net would mean that the model complexity reduction by regularization might be too substantial.

4.2 Hyperparameter Tuning

Based on the initial performance metrics, we've established the need for further tuning and handling of data so that the model can capture and

generalize data more efficiently, which is why we've experimented with the alpha and l1 ratio of the models. In the earlier experiment, we used three alpha values for manual tuning, which were 50, 5, and 0.01. We used an l1 ratio of 0.5 for Elastic Net to keep the balance between the l1 and l2 regularization.

Alpha Value: High = 50:

As with the alpha value set high for Ridge Regression, its performance was slightly better in terms of both MSE and R^2 than Linear Regression, which means that it is more efficient when regularization is strong. However, Lasso and Elastic Net showed high MSE and negative R^2, which indicated an over-regularization of the model.

Alpha Value: Mid = 5

All models achieved an increase in performance with the given mid-range alpha value. The performance of Ridge Regression was similar to that of Linear Regression for MSE and R^2, but Lasso and Elastic Net were better than the high alpha setting. The high Elastic Net R squared value implies a more balanced regularization term

Alpha Value: Low = 0.01

The models performed similarly at a low alpha value, with Linear Regression, Ridge Regression, and Lasso exhibiting comparable metrics. Elastic Net also performed well, with MSE and R^2 aligning closely with Linear Regression, indicating less impact from regularization. This makes sense because the closer the alpha value is to 0, the less regularization it has, which is almost the same as the base linear regression model.

Alpha Settings for Each Model

Ridge Regression demonstrated the best performance with a mid-range alpha value, showing improvements in MSE, R^2, and MAPE compared to other settings. This suggests that moderate regularization strikes an effective balance between fitting the data and avoiding overfitting. In contrast, Lasso and Elastic Net experienced significant performance degradation at high alpha values, indicating that excessive regularization can overly constrain these models, mainly when dealing with multicollinear data. However, Elastic Net achieved the highest R^2 with a balanced alpha value, highlighting its ability to combine L1 and L2 regularization effectively. These results underscore the importance of carefully adjusting alpha values, with mid-range values generally providing the optimal trade-off between regularization strength and model fit.

Higher alpha values do not work well for Lasso and Elastic Net due to their nature of zeroing out coefficients. This leads to solid regularization and degraded performance. However, this is fine with Ridge regression since it does not force them to zero, which avoids issues when multicollinearity is present.

GridSearchCV

GridSearchCV systematically explored a predefined set of hyperparameters using an exhaustive search combined with cross-validation. The primary evaluation metric used was the negative mean squared error (MSE), which allowed the algorithm to select the model configuration that minimized error while maximizing predictive accuracy.

The Ridge model, tuned with an alpha of 10.0, showed the best performance, with the lowest MSE (20.8471) and the highest R^2 (0.7171). The Lasso and ElasticNet models performed similarly, with very close MSE and R^2 values. The selected alpha values suggest a small amount of regularization to improve generalization without overfitting. ElasticNet, which combines both L1 and L2 regularization, offered a slight edge in flexibility, though its improvement over Lasso was marginal. For this hypertuning method, the ridge model emerged as the best-performing model.

RandomizedSearchCV

Another method employed was the RandomizedSearchCV, which performs hyperparameter tuning by sampling a specified number of parameter combinations from a defined distribution rather than exhaustively exploring all possible combinations as in GridSearchCV.

The Elastic Net model, tuned with an alpha of 0.2158 and an l1_ratio of 0.9729, showed the best performance using RandomizedSearchCV, with the lowest MSE (20.8063) and the highest R^2 (0.7177). The Ridge and Lasso models had comparable MSE and R^2 values, but ElasticNet's combination of L1 and L2 regularization provided a slight advantage in predictive accuracy. The selected parameters indicate a balanced regularization approach, improving generalization without overfitting. Elastic Net emerged as the best-performing model for this tuning method since it can give more specific values for the l1 ratio.

4.3 Outlier Removal

Upon data analysis, we observed that outliers can significantly affect the performance of regression models. This is why I also conducted two experiments about this factors, which answered if it's necessary or

not and if it should be done before or after the data split:

Note that this experiment used RM, LSTAT, and PTRATIO as the predictors. The skewness seen in the data analysis part was addressed through log transformation.

Necessary or Unnecessary

The results show that removing outliers led to improved performance metrics, indicating that outliers had a negative impact on the models. This reinforces the importance of considering outlier removal, as it improves MSE, R^2, and MAPE values across all models. But for this case outlier removal was only done to RM and LSTAT, since PTRatio had a skewed distribution even after treatment, removing outliers may be detrimental to the performance as it might take away important data points.

Removing Outliers Before or After Data Split

To assess the impact of data splitting on model performance, I compared scenarios where outlier removal was performed before and after splitting the dataset. This comparison highlights the issue of data leakage. Here is a table of the test performance metric and training performance metric to further show the possibility of data leakage:

Model	MSE	R^2	Adj R^2	MAPE
Ridge	23.9352	0.7171	0.7149	0.1769
Lasso	24.3307	0.7124	0.7102	0.1775
Elastic Net	25.4830	0.6988	0.6965	0.1817

Result of Treating Before Splitting (Training)

Model	MSE	R^2	Adj R^2	MAPE
Ridge	10.9181	0.8331	0.8279	0.1404
Lasso	10.9321	0.8329	0.8277	0.1404
ElasticNet	10.8946	0.8334	0.8283	0.1404

Result of Treating Before Splitting (Test)

When outlier removal is performed before data splitting, the training performance metrics (MSE, R^2, Adj R^2) show significantly lower values, while the test performance metrics are notably higher. This disparity indicates that the models, having access to the entire dataset, benefit from outlier removal but suffer from data leakage. The inflated test results suggest that the model's performance is overestimated because it has seen the entire dataset during preprocessing.

Model	MSE	R^2	Adj R^2	MAPE
Ridge	21.5786	0.7430	0.7410	0.1730
Lasso	21.9724	0.7383	0.7363	0.1732
Elastic Net	23.1117	0.7247	0.7226	0.1770

Result of Treating After Splitting (Training)

Model	MSE	R^2	Adj R^2	MAPE
Ridge	20.8471	0.7171	0.7084	0.1519
Lasso	20.9513	0.7157	0.7069	0.1528
ElasticNet	20.9154	0.7162	0.7074	0.1525

Result of Treating After Splitting (Test)

In contrast, when outlier removal is performed after splitting the data, the performance metrics for both training and test sets are more consistent. The training metrics are higher than the test metrics, which is expected as the model is trained on a subset of data and evaluated on a separate test set. This approach avoids data leakage, providing a more accurate measure of model performance.

4.4 Multicollinearity

Setup 1

Using all predictors yields the best R^2 and MSE results, suggesting a better model fit. However, the high Variance Inflation Factors (VIFs) for several predictors, such as TAX and RAD, indicate significant multicollinearity. This high R^2 is inflated and unreliable, reflecting the instability of coefficient estimates due to multicollinearity.

Setup 2

Removing predictors with high VIF values while keeping the predictors with VIF > 5 (TAX and RAD) shows a decrease in R^2 but a more realistic model performance. The VIF values for TAX and RAD decrease significantly, indicating that their impact on multicollinearity is mitigated when other predictors are removed.

Setup 3

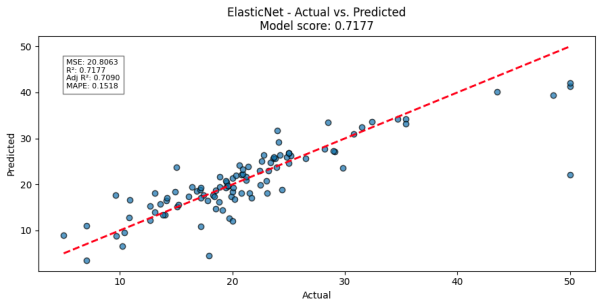
Using only directly correlated predictors (RM, LSTAT, and PTRATIO) results in lower VIF values close to 1. This indicates minimal multicollinearity and suggests that the remaining predictors have little impact on each other. Although this setup shows the lowest MSE and highest precision in coefficient estimation, the performance metrics are somewhat lower than the different setups.

The results highlight the issues caused by multicollinearity, such as inflated R^2 values and unreliable coefficient estimates. Setup 1 demonstrates how multicollinearity can lead to misleadingly high model performance metrics.

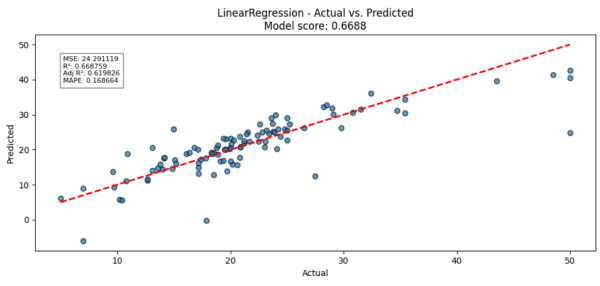
4.5 Final Performance Metrics

To make the summary of this entire section organized and to finalize the results. Through various experiments of treatment and combination, the best model produced by this paper is the Elastic Net Regression Model that uses an alpha value of 0.2158 and an l1 ratio of 0.9729 which was found using RandomizedSearchCV tuning.

	MSE	R2	Adj R2	MAPE
Elastic Net	20.8063	0.7177	0.7090	0.1518



Highest Performing Model: Elastic Net Regression



ElasticNet balances L1 and L2 regularization, which provides a slight edge in predictive accuracy and generalization, making it the most effective model according to the given metrics.

V. Conclusions

5.1 Objectives Review

The primary goal of this activity was to develop a predictive model to estimate median house prices using neighborhood statistics. Through systematic approach we have achieved the following objectives:

Exploratory Data Analysis (EDA)

Comprehensive analysis and visualization of data revealed key relationships between predictors and target variables (MEDV). We could see potential features, patterns, and problems through understanding the dataset's structure. Outliers, non-normal distributions, extreme values, null values, correlations, multivariate relationships, and other data information were extracted.

Data Preprocessing

Data abnormalities, outliers, skewness and multicollinearity were addressed through various preprocessing techniques. Specifically:

- Feature selection was done through coefficient selection, where LSTAT, RM and PTRATIO were selected for having a correlation > +-.50. This helps avoid multicollinearity, lessen noise and make patterns easier to recognize for regression models.
- Data Splitting was done to have test and training data sets and to avoid data leakage.
- Data skewness for LSTAT was treated. Using Log transformation.
- Outlier removal using Z-score based removal was done to features with normal distribution which were RM and LSTAT.
- Creating new variables from pre-existing variables which helped avoid multicollinearity.
- Data scaling and normalization were applied to ensure models were trained on properly formatted data.

Model Implementation and Tuning

Multiple regressions were implemented: Linear Regression, Ridge Regression, Lasso

Regression, and Elastic Net Regression. Hyperparameter tuning was conducted using GridSearchCV and RandomizedSearchCV to optimize model performance.

During this stage, the weaknesses of linear regression to noise, multicollinearity, and weak predictive power were found. The ElasticNet model, with its balanced regularization approach, emerged as the best-performing model across various metrics.

Model Evaluation and Comparison

Performed model evaluation such as MSE, R^2, adjusted-R2 and MAPE. Additionally, scatter plot graphs were included for visual comparison. Elastic Net was the top performer in most cases and demonstrated both consistent better prediction accuracy properties over other models.

Feature Importance & Regularization:

Regularization effects were studied on feature selection and coefficient estimation. Combining L1 and L2 penalties in Elastic Net worked well to address multicollinearity as part of regularization which allowed the model to converge more gently toward a solution, by zeroing out features that would otherwise be useful but were not captured due to interactions.

5.2 Additional Findings

Problem of Underfitting and Overfitting

During the experimentation, underfitting and overfitting became evident and easy to see due to the nature of the regression models and the data set. Underfitting occurred the most when the model was overly regularized, which made it too simplistic. Overfitting happened when the model used all of the predictors, which captured noises and fluctuations in the training data, which led to poor performance on the test data.

Importance of Regularization (Loss Functions)

Regularization is essential for improving the performance of different models, and it helps solve issues like overfitting, multicollinearity, etc. Ridge regression is a linear model that only applies the L2 regularization, allowing us to avoid multicollinearity. Lasso regression, the L1 regularization technique, not only helps in feature selection but can be used to reduce some coefficients (i.e., feature weights) to zero and also makes the model simple. A disadvantage of lasso with multicollinearity may cause under-fitting. ElasticNet is an extension of Lasso and Ridge Regression methods that combines both l1 + l2 regularizations and penalties to provide a more flexible approach from either feature selection. Understanding how the regularization works for each model allows us to find parameters that can lead to optimal models.

Importance of Data Preprocessing

Data preprocessing, including outlier removal, feature selection, transformation, and feature engineering, significantly impacts model performance. Outliers, if not processed, can distort model training and evaluation. Additionally, handling skewed distributions through transformations (e.g., log transformation) can enhance the effectiveness of regression models by making the data more normally distributed, which is beneficial for models assuming linear relationships. The more sensitive the model is to relationships, the more the data should be appropriately handled.

Final Result

A model was created that predicts the median value of owner-occupied homes in Boston (MEDV) using the Boston house-price dataset. Particularly the model created was an Elastic Net Model with an R2 score of 0.7177 and an MSE of 20.8063 which means

5.3 Recommendations

Enhance Feature Engineering

It was shown earlier that despite attempts to create additional variables, it did not prove to have much of an improvement. Integrating polynomial features or increasing dimensionality of the model may improve the model. Approximately 71.77% of the variance in the median house prices is explained by the model and on average, the squared difference between the predicted and actual median house prices is about 20.81.

Improve Data Preprocessing

The methods used in this paper were insufficient, considering that there are more advanced methods to preprocess data, such as advanced imputation methods to handle outliers or missing data or DBSCAN to better identify outliers.

Use Robust Regressions or More Advanced Models

Robust regressions can handle outliers and leverage data better than standard regression methods. They provide more accurate and reliable estimates.

VI. References

Wilhelm, Jochen. (2015). Re: Which is the best method for removing outliers in a data set?. Retrieved from: https://www.researchgate.net/post/Which-is-the-best-method-for-removing-outliers-in-a-data-set/565ca78f64e9b26a9c8b45c5/citation/download.

Daniëls, H., Kamp, B., & Verkooijen, W. Application of Neural Networks to House Pricing and Bond Rating. https://core.ac.uk/download/pdf/6794798.pdf

Strum, R., Mowbray, F., Zargoush, M., & Jones, A. (2023). Prehospital prediction of hospital admission for emergent acuity patients transported by paramedics: A population-based cohort study using machine learning. PLoS One, 18(8), e0289429.

Kumar, S., Attri, S. D., & Singh, K. K. (2019). Comparison of Lasso and stepwise regression technique for wheat yield prediction. Journal of Agrometeorology, 21(2), 188-192.

Multicollinearity in Regression Analysis: Problems, Detection, and Solutions - Statistics By Jim. https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/

Should Feature Selection be done before Train-Test Split or after? (n.d.). Stack Overflow. https://stackoverflow.com/questions/56308116/should-feature-selection-be-done-before-train-test-split-or-after

Kfollis. (2023, October 31). Feature selection (Data mining). Microsoft Learn. https://learn.microsoft.com/en-us/analysis-services/data-mining/feature-selection-data-mining?view=asallproducts-allversions

Poon, W. (2022, March 19). Feature Engineering for Machine Learning (2/3) - towards data science. Medium. https://towardsdatascience.com/feature-engineering-for-machine-learning-434c9b4912c6

Patel, H. (2024, April 29). Feature Engineering explained. Built In. https://builtin.com/articles/feature-engineering

Kumar, A. (2023, May 14). Boston Housing Dataset Linear Regression: Predicting House Prices - Analytics Yogi. Analytics Yogi. https://vitalflux.com/boston-housing-dataset-linear-regression-predicting-house-prices/

Is there such a thing as a too low R-squared when running multiple linear regression? (n.d.). Cross Validated. https://stats.stackexchange.com/questions/509933/is-there-such-a-thing-as-a-too-low-r-squared-when-running-multiple-linear-regres