PROBLEM 5

HOUSING PRICE IN BOSTON

ZHU XIAOYANG DC125693
YANG YISHU DC128280
TANG YUYAN DC126862

TABLE OF CONTENTS

01

INTRODUCTION

- Problem
- Variable explanation

1.1 PROBLEME

Problem 5 (Lasso regression)**.** Some factors are thought to influence the housing price in Boston. In this project, we will investigate which of those are more important by using Lasso regression. For this, one needs to:

- Find the dataset of this project here, split it into two datasets for training and testing respectively, do other necessary pre-processing based on descriptive statistical analysis.
- Build a proper Lasso regression model as learned in Lecture 2 and 4.
- Use the data in training set to estimate the model you build in last step by coding via R or Python, here using packages are allowed.
- Evaluate the model estimated on last step by computing the mean squared error with using data on test set.
- Present and explain the result.

1.2 VARIABLE EXPLANATION

- **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; else 0)**
- **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 centres**
- **RAD – index of accessibility to radial highways**
- **TAX – full-value property-tax rate per 10,000 dollar**
- **PTRATIO – pupil-teacher ratio by town**
- **B – 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town**
- **LSTAT – % lower status of the population**
- **MEDV – Median value of owner-occupied homes in (dollar) 1000's**

02

DATA PRE-PROCESSING

2.1 AN OVERALL GLANCE

Apparent Peculiar Patterns:

1. For feature “zn”:
   Q1=Q2=0 nonzero values may centered above 12.5

2. For feature “chas”:
   Q1=Q2=Q3=0, yet max=1, it is a encoded categorical variable

	count	mean	std	min	25%	50%	75%	max
crim	506.0	3.613524	8.601545	0.00632	0.082045	0.25651	3.677083	88.9762
zn	506.0	11.363636	23.322453	0.00000	0.000000	0.00000	12.500000	100.0000
indus	506.0	11.136779	6.860353	0.46000	5.190000	9.69000	18.100000	27.7400
chas	506.0	0.069170	0.253994	0.00000	0.000000	0.00000	0.000000	1.0000
nox	506.0	0.554695	0.115878	0.38500	0.449000	0.53800	0.624000	0.8710
rm	506.0	6.284634	0.702617	3.56100	5.885500	6.20850	6.623500	8.7800
age	506.0	68.574901	28.148861	2.90000	45.025000	77.50000	94.075000	100.0000
dis	506.0	3.795043	2.105710	1.12960	2.100175	3.20745	5.188425	12.1265
rad	506.0	9.549407	8.707259	1.00000	4.000000	5.00000	24.000000	24.0000
tax	506.0	408.237154	168.537116	187.00000	279.000000	330.00000	666.000000	711.0000
ptratio	506.0	18.455534	2.164946	12.60000	17.400000	19.05000	20.200000	22.0000
b	506.0	356.674032	91.294864	0.32000	375.377500	391.44000	396.225000	396.9000
lstat	506.0	12.653063	7.141062	1.73000	6.950000	11.36000	16.955000	37.9700
medv	506.0	22.532806	9.197104	5.00000	17.025000	21.20000	25.000000	50.0000

2.2 DATA PRE-PROCESSING

2.2.1 Check whether there is any missing values in the dataset.

```
missing_info= round(housing.isna().sum() * 100/housing.shape[0], 2)
missing_info
```

```
crim       0.0
zn         0.0
indus      0.0
chas       0.0
nox        0.0
rm         0.0
age        0.0
dis        0.0
rad        0.0
tax        0.0
ptratio    0.0
b          0.0
lstat      0.0
medv       0.0
dtype: float64
```

```
# No missing value exists
```

Luckily, there is no missing values in the dataset.

2.2 DATA PRE-PROCESSING

2.2.2 Check whether all the data are numerical data.

```
housing.info()
```

```
<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    int64  
 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(11), int64(3)
memory usage: 55.5 KB
```

Luckily, all the data are numerical.

2.2 DATA PRE-PROCESSING

2.2.3 Check whether there is any skewness in the house price variable.

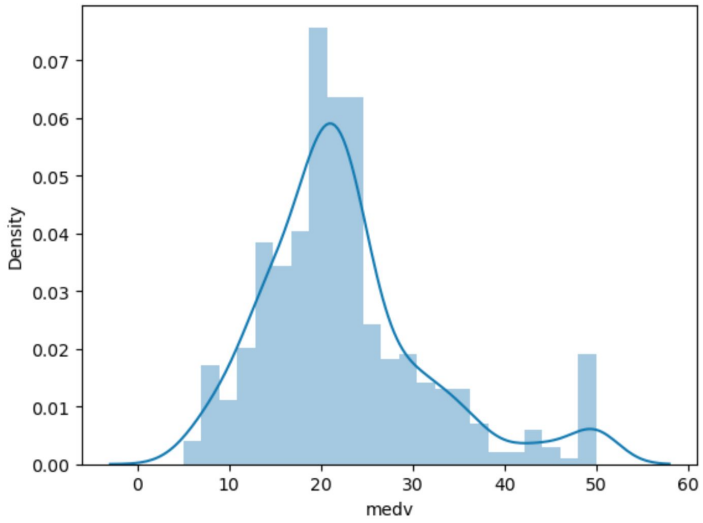
Q: Why should we check skewness of the dependent value?

A: Although Lasso can handle skewed data, highly skewed data may affect the performance of the model.

2.2 DATA PRE-PROCESSING

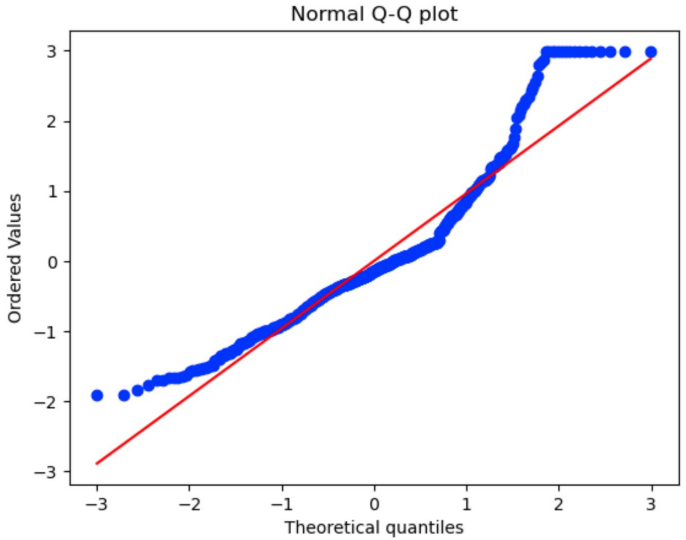
2.2.3 Check whether there is any skewness in the house price variable.

```
: sns.distplot(housing['medv'])
: <Axes: xlabel='medv', ylabel='Density'>
```




```
z = (housing['medv']-np.mean(housing['medv']))/np.std(housing['medv'])
stats.probplot(z, dist="norm", plot=plt)

plt.title("Normal Q-Q plot")
plt.show()
```



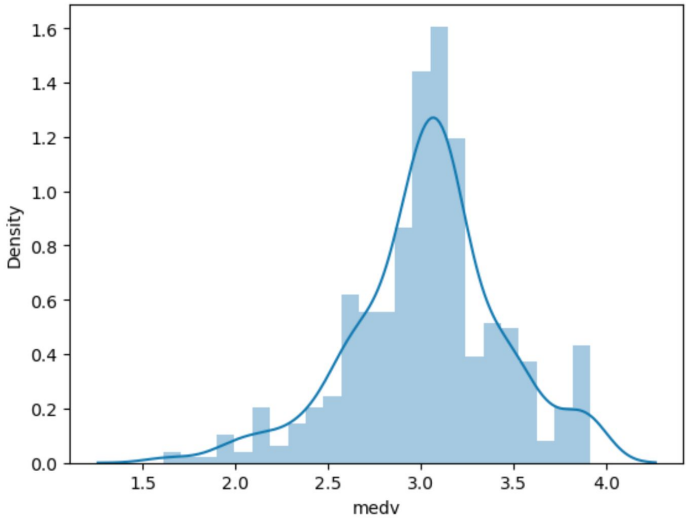
Moderately skewed. Improvements are needed.

2.2 DATA PRE-PROCESSING

2.2.3 Check whether there is any skewness in the house price variable.

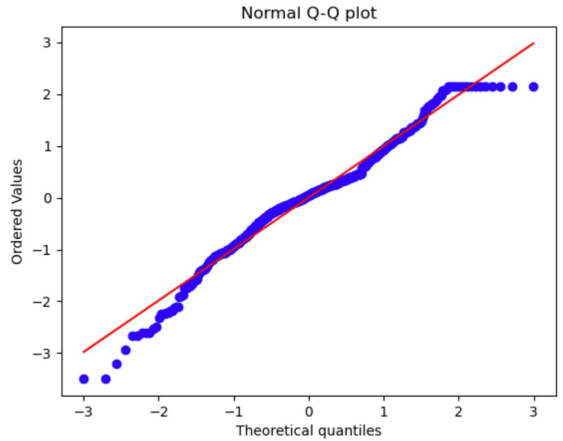
We chose log transformation to remove the skewness.

```
sns.distplot(np.log(housing['medv']))
plt.show()
```




```
z = ((np.log(housing['medv']))-np.mean(np.log(housing['medv'])))/np.std((np.log(housing['medv'])))
stats.probplot(z, dist="norm", plot=plt)

plt.title("Normal Q-Q plot")
plt.show()
```

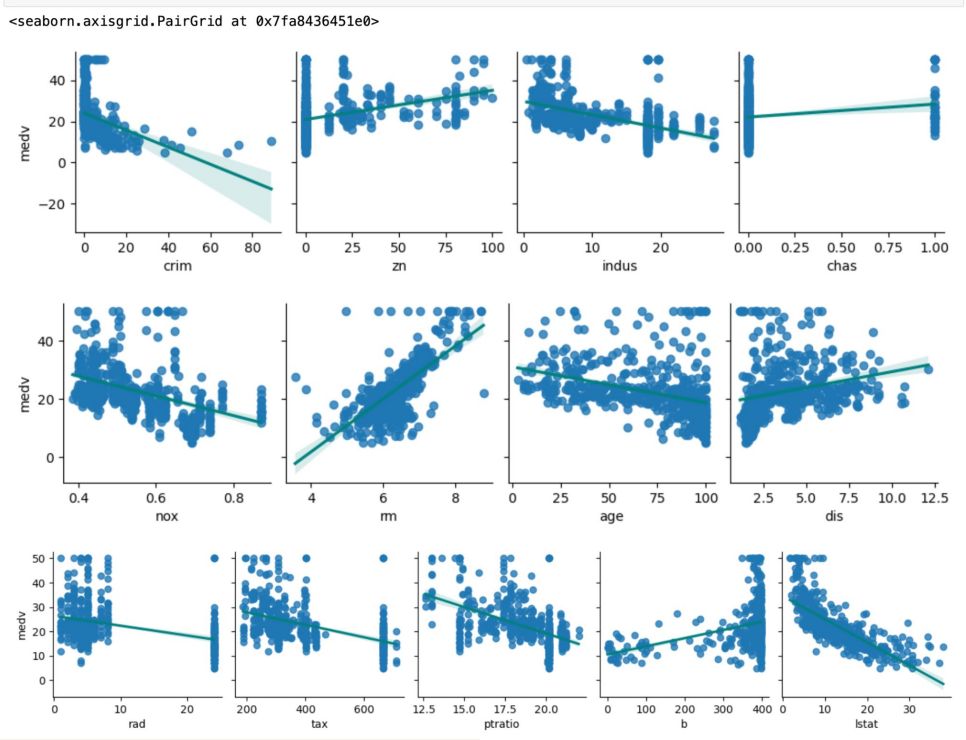


Now, the dependent variable is close to the Normal distribution.

2.2 DATA PRE-PROCESSING

2.2.4 Check and adjust the outliers in each exploratory variables

· An overall glance: plot each exploratory variable against house price

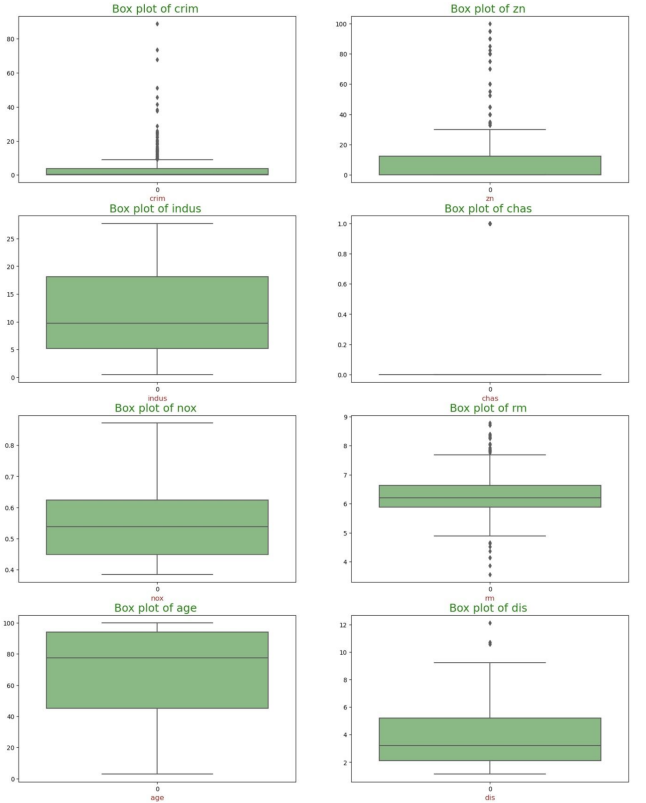


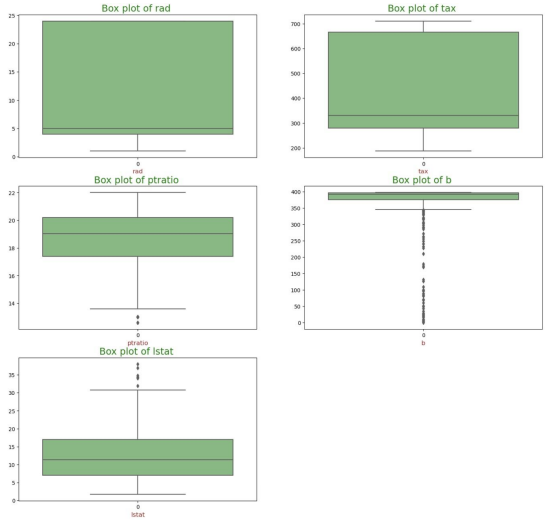
· Some features are observed to have outliers.

· We need box plot to tell us there are truly outliers.

2.2 DATA PRE-PROCESSING

2.2.4 Check and adjust the outliers in each exploratory variables





· We can theoretically discover that there are outliers in features ‘crim’, ‘zn’, ‘rm’, ‘dis’, ‘ptratio’, ‘b’, and ‘lstat’.

· Adjustments are needed.

2.2 DATA PRE-PROCESSING

2.2.4 Check and adjust the outliers in each exploratory variables

```
num_cols= housing.columns.tolist()
num_cols.remove('medv_log_trans')
num_cols.remove('medv')
scaler= RobustScaler(quantile_range=(2, 98))
scaler.fit(X_train[num_cols])
X_train[num_cols]= scaler.transform(X_train[num_cols])
X_test[num_cols]= scaler.transform(X_test[num_cols])
```

· Thanks to Robust Scalar().

Data are rescaled in a robust way.

· Outliers will no longer bother us.

After robust rescaling:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
5	-0.009315	0.0	-0.314168	0.0	-0.169873	0.060810	-0.206777	0.390520	-0.086957	-0.228330	0.012195	0.006725	-0.207001
116	-0.005134	0.0	0.013387	0.0	0.019111	-0.015465	-0.053773	-0.049851	0.043478	0.215645	-0.097561	0.004537	0.040386
45	-0.003497	0.0	-0.116297	0.0	-0.191107	-0.163811	-0.482848	0.263408	-0.086957	-0.205074	-0.085366	0.014143	-0.025898
16	0.032773	0.0	-0.064842	0.0	0.000000	-0.087836	-0.532741	0.183874	-0.043478	-0.048626	0.292683	-0.012676	-0.157378
468	0.629592	0.0	0.351818	0.0	0.089183	-0.090539	-0.070404	-0.026287	0.826087	0.710359	0.195122	-0.061004	0.260969

2.2 DATA PRE-PROCESSING

2.2.5 Split Data into training and testing sets

```
y= housing['medv_log_trans']
X= housing.drop('medv_log_trans',axis=1)
X= X.drop('medv',axis=1)

X_train, X_test, y_train, y_test= train_test_split(X, y, train_size= .7, random_state= 42)
```

· We put 70% of data into training set and the remaining 30% to the testing set.

```
X_train.shape
```

```
(354, 13)
```

```
X_test.shape
```

```
(152, 13)
```

```
y_train.shape
```

```
(354,)
```

```
y_test.shape
```

```
(152,)
```

03

METHOD

- Model Explanation: Lasso

3.1 LASSO BACKGROUND: MULTIVARIATE LINEAR REGRESSION

The multiple linear model can be represent as:

$$Y = f(X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$$

Where $X = \left[x_1, x_2, \ldots, x_p\right] \in R^{n \times p}$ and Y is the output, also we have ε as the error term.

The sample contains **N** cases,

each of which consists of p covariates and a single output.

- Output is subject to $X = \left[x_1, x_2, \ldots, x_p\right] \in R^{n \times p}$
- It is determined when $\beta_0, \beta_1, \ldots, \beta_p$ are chosen
- β_0 is the constant coefficient

Lasso Regression Model

So the meaning of Lasso Regression model is to minimize the following error with specific restriction.

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{N}(y_i-\beta_0-x_i\,\beta)^2\right\}$$ ***object:*** with restriction: $$\sum_{j=1}^{p}|\beta_j|\le t.$$

Here the sample contains **N** cases, each of which has p covariate and a specific outcome.

$$y_i \text{ be the outcome} \qquad x_i:=\left(x_1,x_2,\ldots,x_p\right)_i \text{ be the covariate vector for the } i^{th} \text{ case}$$

Here t is the K we refer to in the lasso regression model

Letting X be the covariate matrix, so that $X_{ij}=(x_i)_j$ and x_i is the i^{th} row of X, the expression can be written more compactly as

$$\min_{\beta_0,\beta}\left\{\|y-\beta_0-X\beta\|_2^2\right\} \text{ subject to } \|\beta\|_1\le t,$$

Lagrangian multiplier:

$$L\left(\beta_0, \beta, \lambda\right) = ||y - \beta_0 - X\beta||^2 + \lambda\left(||\beta||_1 - K\right)$$

Positive turning parameter to be chosen: λ

By doing **(1)** and finding **minimizer by differentiation**, we have:

$$\hat{\beta} = (X^T X)^{-1} X^T \left(y - \beta_0\right) - \frac{\lambda}{2}(X^T X)^{-1} sgn(\hat{\beta})$$

Where $X^T X$ should be invertible.

$$sgn(\beta) = \left[sgn\left(\beta_1\right), \ldots, sgn\left(\beta_p\right)\right]^T$$

$$sgn(x) = \begin{cases} 1, x > 0 \\ -1, x < 0 \\ random\ value\ in\ [-1, 1], x = 0 \end{cases}$$

KKT Condition:

$$1. \nabla_\beta L\left(\beta_0, \beta, \lambda\right) = 0$$

$$2. ||\beta||_1 \leq K$$

$$3. \lambda \geq 0$$

$$4. \lambda\left(||\beta||_1 - K\right) = 0$$

Relation between λ *and* K

By Employing **(3)** and **(4)** of **KKT** Conditions. → $3.\lambda \geq 0$, $4.\lambda\left(\|\beta\|_1 - K \right) = 0$

$$\hat{\beta} = (X^T X)^{-1} X^T \left(y - \beta_0 \right) - \frac{\lambda}{2} (X^T X)^{-1} sgn(\hat{\beta}) = \hat{\beta}^{ls} - \frac{\lambda}{2} (X^T X)^{-1} sgn(\hat{\beta})$$

where $\hat{\beta}^{ls} = (X^T X)^{-1} X^T \left(y - \beta_0 \right)$

$\lambda = 0 \Leftrightarrow$ *restriction has no effect*

By KKT (4), *we have* $\| \hat{\beta}^{ls} - \frac{\lambda}{2} (X^T X)^{-1} sgn(\hat{\beta}) \|_1 = K \longrightarrow \lambda \uparrow, K \downarrow$

If $\hat{\beta}_j{}^{ls} > 0$: *as* $\lambda > 0$ *is small, we have* $\hat{\beta}_j > 0$.

Let λ *increases then* $\hat{\beta}_j$ *decreases to* 0, *which is also shown by similarity when* $\hat{\beta}_j{}^{ls} < 0$.

As λ *increases again,* $\hat{\beta}_j$ *keeps* 0 *by contradiction*.

for every **j** belongs to the Set: **S={1,2,…,p}**

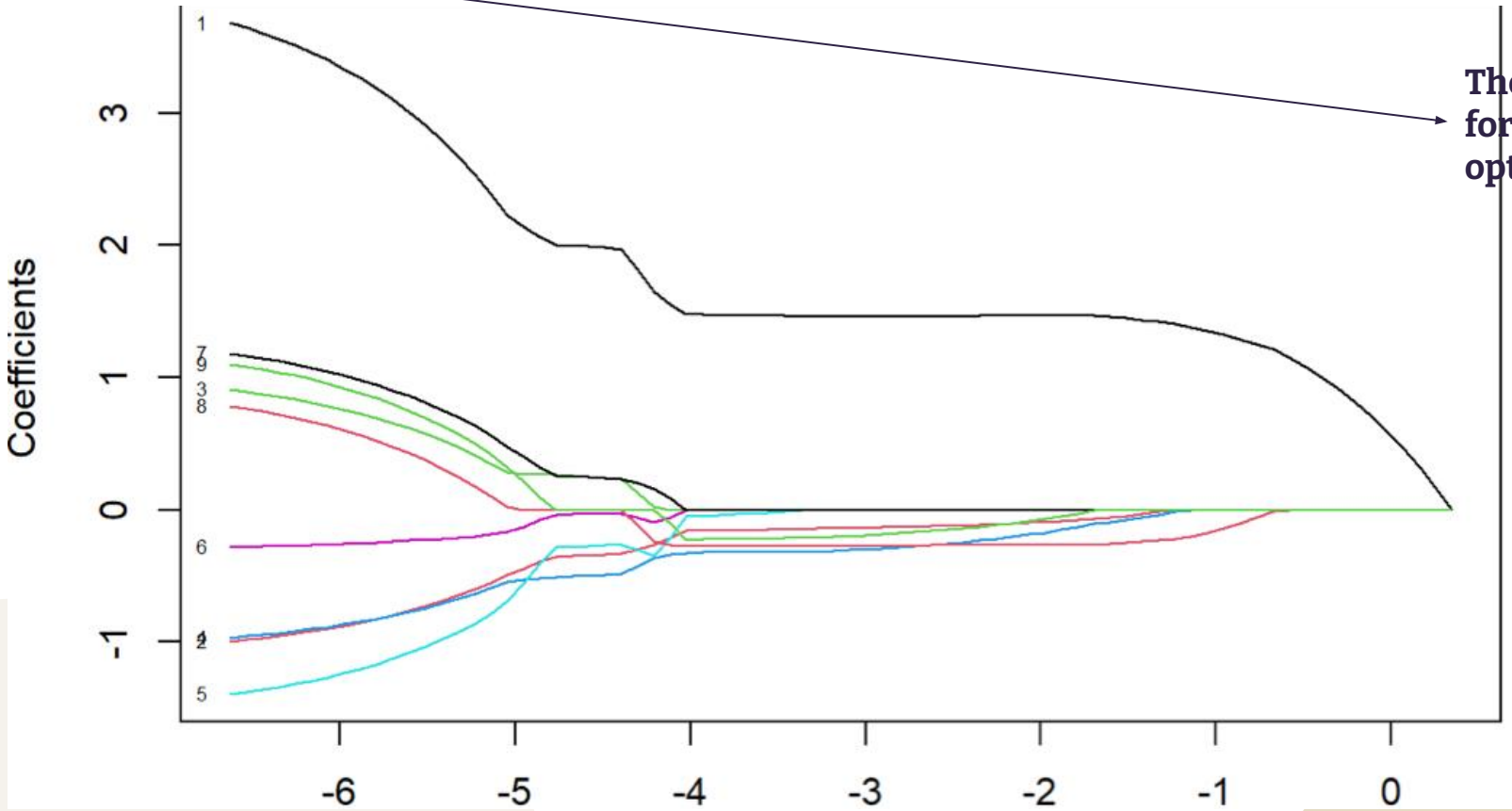
Well it should be noticed that:

$$When\ \widehat{\beta}_j = 0,\ we\ have\ 0 = \widehat{\beta}_j^{\,ls} - \frac{\lambda}{2}(X^T X)^{-1} sgn(0)\ .$$

$$\Rightarrow \lambda \cdot sgn(0) = 2\widehat{\beta}_j^{\,ls}(X^T X)$$

$$so\ sgn(0)\ \ is\ the\ random\ value\ that\ make\ the\ above\ equation\ satisf\ y$$

So some covariates may decrease to 0 and keep being 0 when we are finding optimal lamda by choosing larger lamda



There is a range for finding optimal lamda

04

MODEL BUILDING & RESULTS

4.1 MODEL BUILDING

4.1.1 Find the optimal lamda

Roughly, which interval in [0, 1] would lamda falls into?

GridSearchCV() is helps us to find the relatively optimal lamda in range 1 by Cross Validation.

```
range1= [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0] # an array of alpha candidates
params_grid= {'alpha': range1} #searching over different values of 'alpha'
lasso= Lasso(random_state= 42)
lasso_gcv= GridSearchCV(estimator= lasso,
                        param_grid= params_grid,
                        cv= 3,
                        scoring= 'neg_mean_absolute_error',
                        return_train_score= True,
                        n_jobs= -1,
                        verbose= 1)

lasso_gcv.fit(X_train, y_train)  #perform the grid search and find the best hyperparameters for the Lasso model on t
```

```
Fitting 3 folds for each of 14 candidates, totalling 42 fits
```

```
lasso_gcv.best_estimator_
```

```
Lasso(alpha=0.001, random_state=42)
```

4.1 MODEL BUILDING

4.1.1 Find the optimal lamda

Optimize the value of lamda by providing finer sub-intervals around 0.001 for GridSearchCV()

```
Fitting 3 folds for each of 18 candidates, totalling 54 fits
```

```
GridSearchCV(cv=3, estimator=Lasso(random_state=42), n_jobs=-1,
             param_grid={'alpha': [5e-05, 6e-05, 7e-05, 8e-05, 9e-05, 0.0001,
                                   0.0002, 0.0003, 0.0004, 0.0005, 0.0006,
                                   0.0007, 0.0008, 0.0009, 0.001, 0.002, 0.003,
                                   0.004]},
             return_train_score=True, scoring='neg_mean_absolute_error',
             verbose=1)
```

▸ estimator: Lasso

▸ Lasso

```
lasso_gcv.best_estimator_
```

```
Lasso(alpha=0.001, random_state=42)
```

The optimal lamda=0.001

4.1 MODEL BUILDING

4.1.2 Fit the data to Lasso Regression model with lamda=0.001

```
lasso_model= lasso_gcv.best_estimator_
lasso_model.fit(X_train, y_train)
```

```
Lasso(alpha=0.001, random_state=42)
```

```
lasso_model.intercept_
```

```
3.1392596093010225
```

```
lasso_model.coef_
```

```
array([-0.23901105,  0.02315828,  0.        ,  0.10211599, -0.21169403,
        0.32481098, -0.00097234, -0.25295107,  0.11991946, -0.08971806,
       -0.27052598,  0.19715086, -0.83781336])
```

The lasso model has been built.

4.2 MODEL EVALUATION

	On training set	On testing set
MSE	0.03588449097029216	0.03838649167639508
R^2	0.7900996167605849	0.7505718946623197
adj.R^2	0.7827130930102243	0.7290385330504336
RMSE	0.1894320220297829	0.19592470920329338
MAE	0.1357048219175345	0.1362031809513098

4.2 MODEL EVALUATION

On testing set, MSE is around 0.0384, RMSE is around 0.1959.

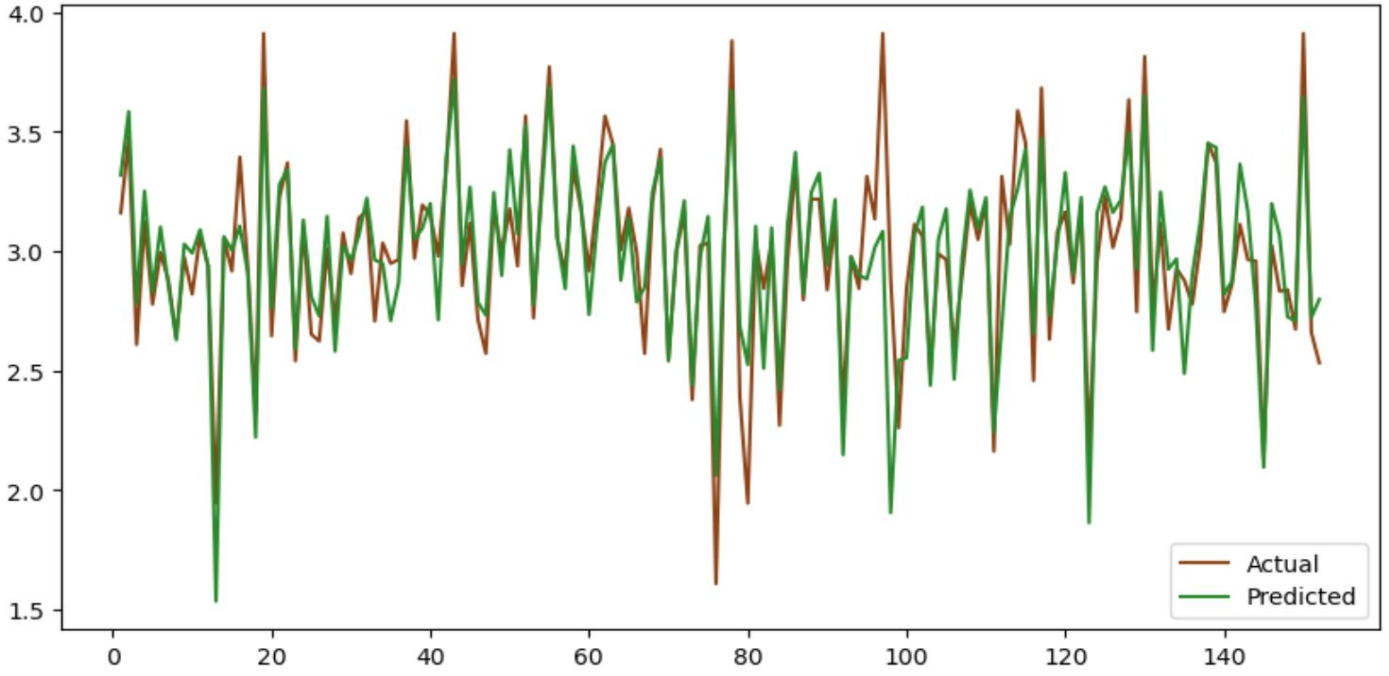
RMSE tells us:

On average, our model's predictions are off by about $195.9 from the actual prices. (Recall our prices are in units of $1000)

The error is tolerable for house price predictions.

4.2 MODEL EVALUATION

Plot the actual and predicted values of the testing set. The curves are very much alike.



4.3 RESULT

4.3.1 More important features

What factors have more significant influences on the housing price in Boston?

```
lasso_coef= pd.Series(lasso_model.coef_, index= X_train.columns)
selected_features= len(lasso_coef[lasso_coef != 0])
print('Features selected by Lasso:', selected_features)
```

```
Features selected by Lasso: 12
```

```
lasso_coef[lasso_coef != 0]
```

```
crim      -0.239011
zn         0.023158
chas       0.102116
nox       -0.211694
rm         0.324811
age       -0.000972
dis       -0.252951
rad        0.119919
tax       -0.089718
ptratio   -0.270526
b          0.197151
lstat     -0.837813
dtype: float64
```

· **indus**–the proportion of non-retail business acres per town was the only factor discarded by Lasso regression

4.3 RESULT

4.3.2 Visualizations of the Result

How to define “more influential”?

· **For the public** – perhaps the top 5 influential features

· **For the government** –perhaps the top 10 influential features

· **For the real estate companies and property investers** –perhaps all influential features selected by Lasso regression

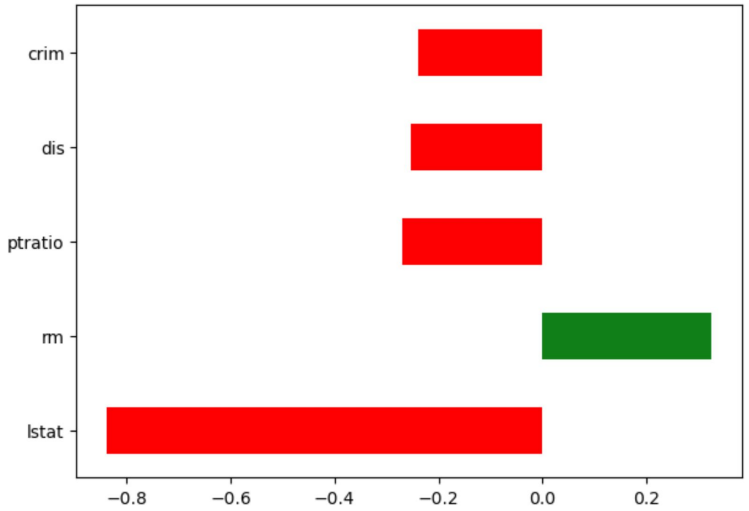
4.3 RESULT

4.3.2 Visualizations of the Result

What factors are more influential on the housing price in Boston?

For people who is only interested in top 5 influential factors:

```
# Top 5
plt.figure(figsize= (7, 5))
top5_features_l.plot.barh(color= (top5_features_l > 0).map({True: 'g', False: 'r'}))
plt.show()
```



- RED-negative effect
- GREEN-positive effect

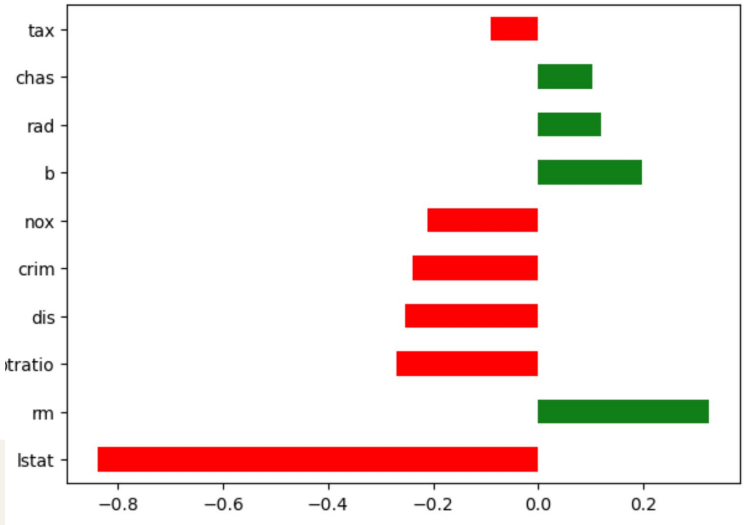
4.3 RESULT

4.3.2 Visualizations of the Result

What factors are more influential on the housing price in Boston?

For people who is interested in top 10 influential factors:

```
Top 10
t.figure(figsize= (7, 5))
p10_features_l.plot.barh(color= (top10_features_l > 0).map({True: 'g', False: 'r'}))
t.show()
```



· RED-negative effect

· GREEN-positive effect

4.3 RESULT

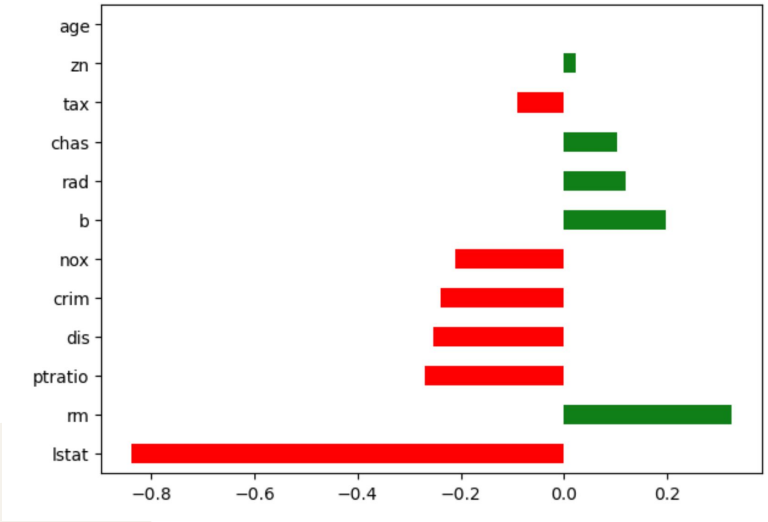
4.3.2 Visualizations of the Result

What factors are more influential on the housing price in Boston?

Visualization of all influential factors selected by Lasso regression:

```
# All 12 selected features
all_features_l=  lasso_coef[abs(lasso_coef[lasso_coef != 0]).nlargest(12).index]

plt.figure(figsize= (7, 5))
all_features_l.plot.barh(color= (all_features_l > 0).map({True: 'g', False: 'r'}))
plt.show()
```



- RED-negative effect
- GREEN-positive effect

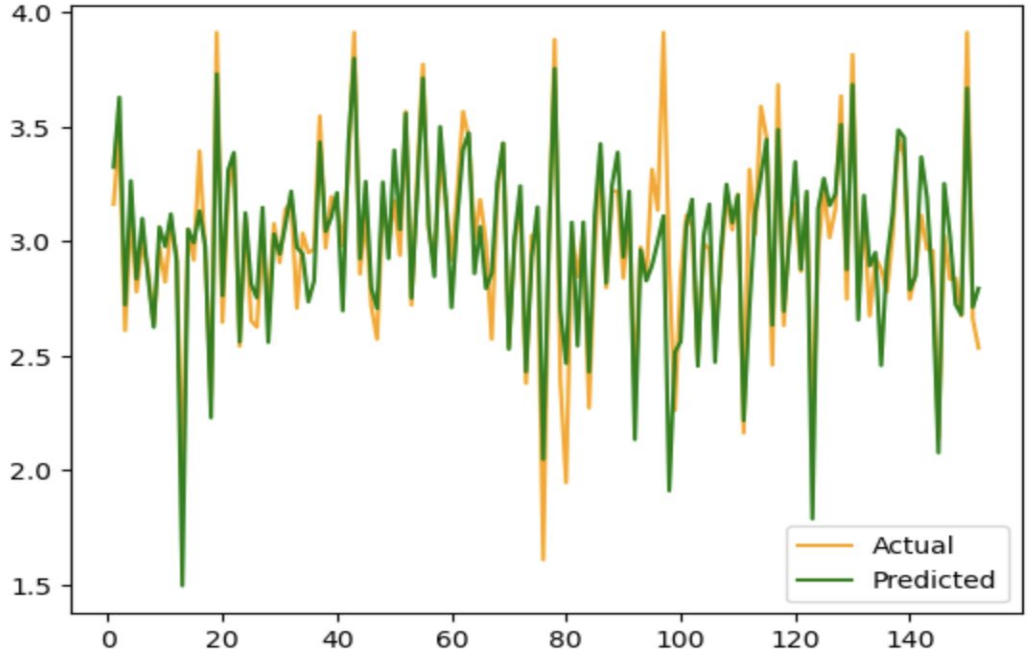
05

FURTHER DISCUSSION

OTHER MODELS ?

5.1.1 MULTIPLE LINEAR REGRESSION

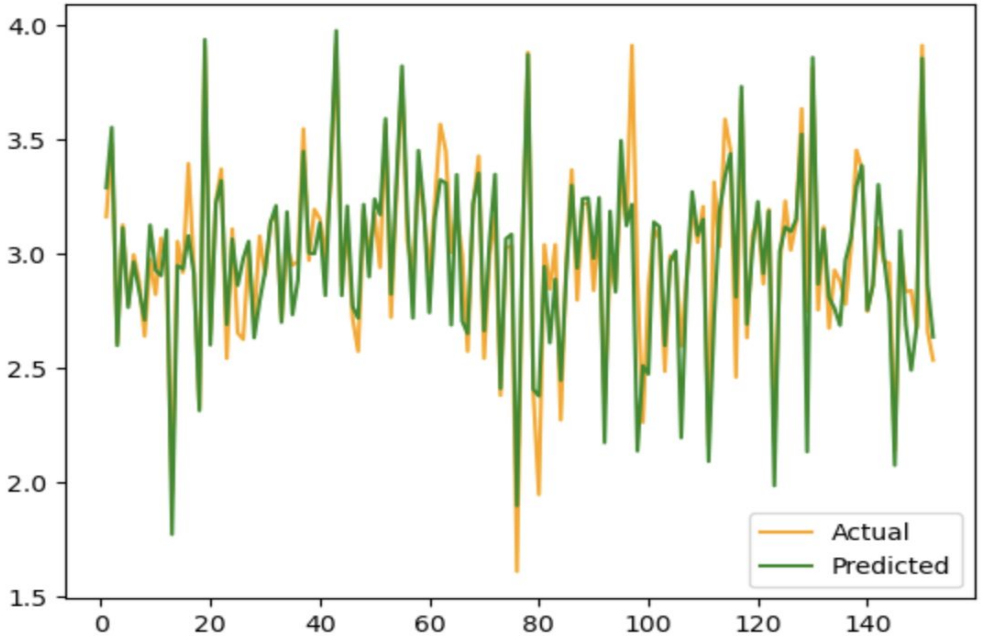
```
plt.figure()
x=np.arange(1,153)
plt.plot(x,y_test,label='Actual',color='orange')
plt.plot(x,y_pred_linear,label='Predicted',color='green')
plt.legend()
plt.show()
```



- **Mean Squared Error: 0.0369**
- **R-squared: 0.7605**
- **Adjusted R-squared: 0.7398**

5.1.2 POLYNOMIAL REGRESSION

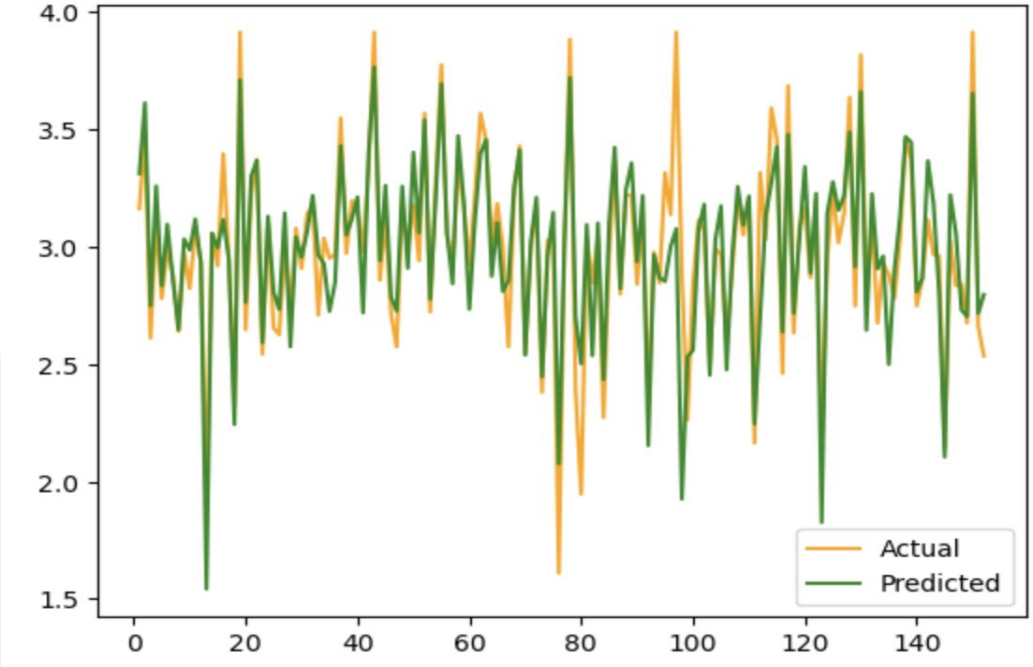
```
plt.figure()
x=np.arange(1,153)
plt.plot(x,y_test,label='Actual',color='orange')
plt.plot(x,y_pred_poly,label='Predicted',color='forestgreen')
plt.legend()
plt.show()
```



- **Mean Squared Error: 0.0312**
- **R-squared: 0.7973**
- **Adjusted R-squared: 0.7798**

5.1.3 RIDGE REGRESSION

```
plt.figure()
x=np.arange(1,153)
plt.plot(x,y_test,label='Actual',color='orange')
plt.plot(x,y_test_pred_r,label='Predicted',color='forestgreen')
plt.legend()
plt.show()
```

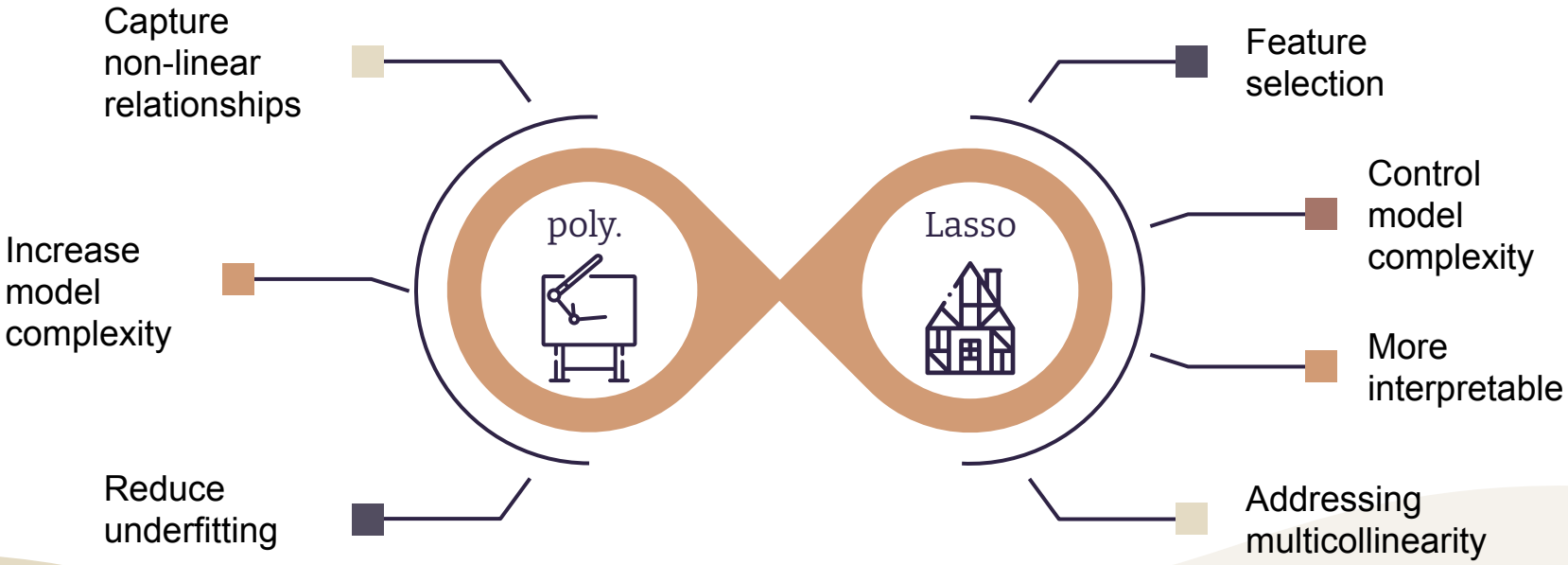


- **Mean Squared Error: 0.0372**
- **R-squared: 0.7582**
- **Adjusted R-squared: 0.7355**

5.1.4 COMPARISON OF FOUR REGRESSION MODELS

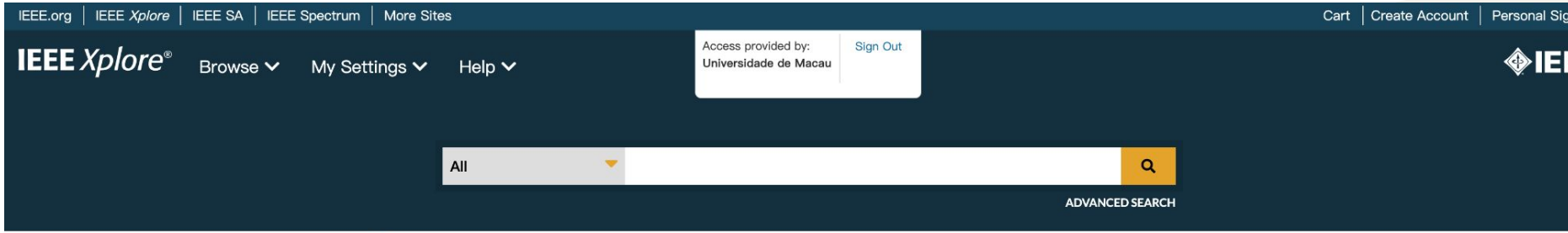
	Details	Mean Square Error (test)	R-squared (test)	Adjusted R-squared (test)
MULTIPLE	/	0.0369	0.7605	0.7398
POLYNOMIAL	degree=2	**0.0312**	**0.7973**	**0.7798**
RIDGE	alpha=1.0	0.0372	0.7582	0.7355
LASSO	alpha=0.001	0.0384	0.7506	0.7290

5.1.5 COMPARISON BETWEEN POLYNOMIAL REGRESSION AND LASSO REGRESSION



TRY TO IMPROVE THE LASSO MODEL

- Dimensionality reduction



Boston House Price Prediction Using Regression Models

Publisher: IEEE

Cite This

PDF

Saptarsi Sanyal ; Saroj Kumar Biswas ; Dolly Das ; Manomita Chakraborty ; Biswajit Purkayastha All Authors

Abstract:

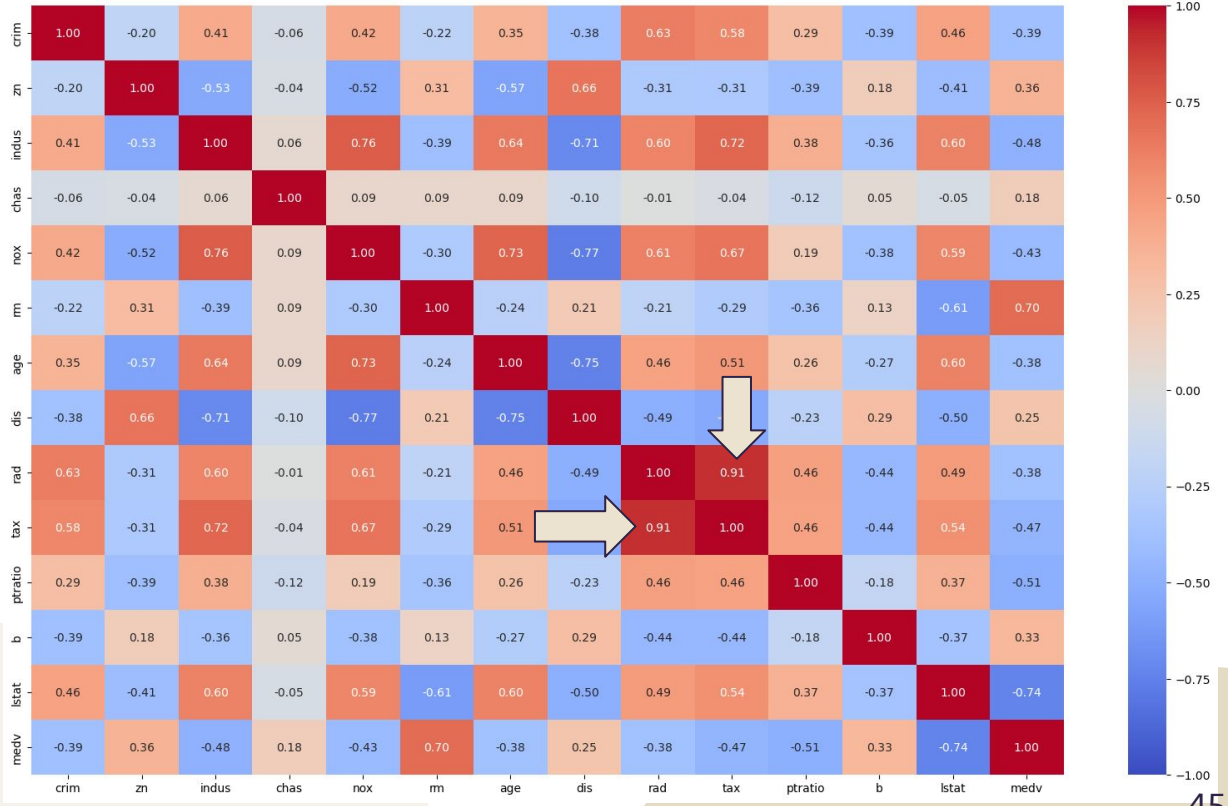
Everyone wishes to buy and live in a house which suits their lifestyle and which provides amenities according to their needs. There are many factors that are to be taken into consideration like area, location, view etc. for prediction of house price. It is very difficult to predict house price as it is constantly changing and quite often the prices are exaggerated for which people who want to buy houses, and various real estate agencies who want to invest in properties, find it difficult to buy or sell houses. For this reason, in this paper the author creates an advanced automated

5.2 TRY TO SIMPLIFY THE MODEL

From heat map it is observed that the attributes **“TAX”** and **“RAD”** are highly correlated (0.91).

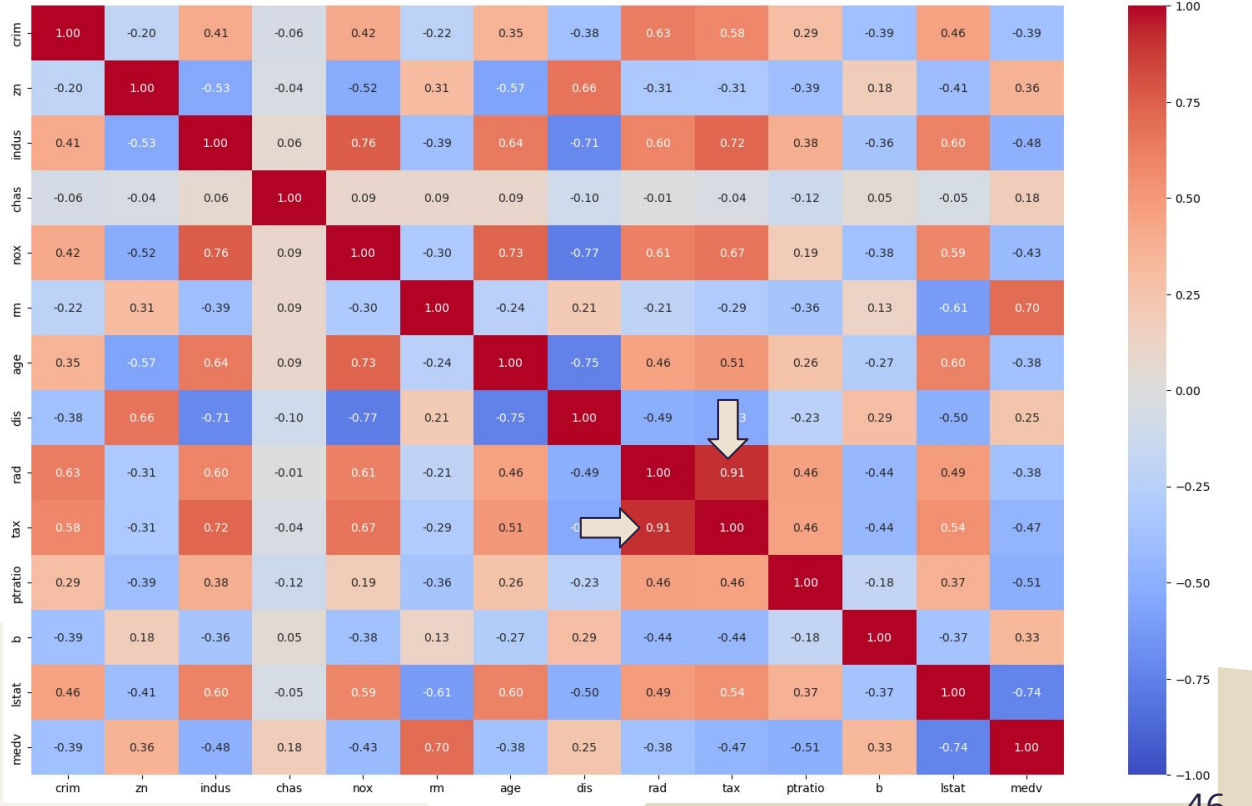
They are having similar behavior and will also have similar impact while doing prediction calculation.

Remove RAD



5.2 TRY TO SIMPLIFY THE MODEL

By calculating the correlation coefficient between the independent variable and the response variable, it can be found that **RM, TAX, NOX, INDUS,PTRATIO, and LSTAT** are six variables that have a greater correlation with the response variable.



5.2 TRY TO SIMPLIFY THE MODEL

Remove insignificant features and check the skewness of the other features.

In [68]:
```
improved = housing.drop(["crim", "zn", "rad", "b","dis","age","chas",'medv_log_trans'], axis=1)
improved.head()
```

Out[68]:

	indus	nox	rm	tax	ptratio	lstat	medv
0	2.31	0.538	6.575	296	15.3	4.98	24.0
1	7.07	0.469	6.421	242	17.8	9.14	21.6
2	7.07	0.469	7.185	242	17.8	4.03	34.7
3	2.18	0.458	6.998	222	18.7	2.94	33.4
4	2.18	0.458	7.147	222	18.7	5.33	36.2

In [69]:
```
skewness = improved.apply(stats.skew)
skewness
```

Out[69]:
```
indus      0.294146
nox        0.727144
rm         0.402415
tax        0.667968
ptratio   -0.799945
lstat      0.903771
medv       1.104811
dtype: float64
```

5.2 TRY TO SIMPLIFY THE MODEL

In [70]:
```
improved['nox_log_trans']= np.log(improved['nox'])
improved['tax_log_trans']= np.log(improved['tax'])
improved['lstat_log_trans']= np.log(improved['lstat'])
improved['medv_log_trans']= np.log(improved['medv'])

improved=improved.drop(['nox','tax','lstat','medv'],axis=1)
```

In [71]:
```
ptratio_boxcox, lam = stats.boxcox(improved['ptratio'])
print("Optimal λ: ", lam)
```

```
Optimal λ:  4.350215431329132
```

Use log transformation and box-cox transformation to fit features to a normal distribution

In [72]:
```
improved['ptratio']=ptratio_boxcox
improved
```

Check the skewness again.

In [73]:
```
skewness = improved.apply(stats.skew)
skewness
```

Out[73]:
```
indus             0.294146
rm                0.402415
ptratio          -0.222262
nox_log_trans     0.356718
tax_log_trans     0.329576
lstat_log_trans  -0.319282
medv_log_trans   -0.329341
dtype: float64
```

5.2 TRY TO SIMPLIFY THE MODEL

Divide into training set（80%）and test set（20%）

In [74]:
```
# According to the article, let 80% be the training set, and 20% be the testing set

y_improved= improved['medv_log_trans']
X_improved= improved.drop('medv_log_trans',axis=1)

X_train_improved, X_test_improved, y_train_improved, y_test_improved= train_test_split(X_improved, y_improved, tr
```

In [75]:
```
X_train_improved.shape
```

Out[75]: (404, 6)

In [76]:
```
X_test_improved.shape
```

Out[76]: (102, 6)

5.2 TRY TO SIMPLIFY THE MODEL

In [80]:
```
params_grid= {'alpha': range2}
lasso_gcv_improved= GridSearchCV(estimator= lasso,
                         param_grid= params_grid,
                         cv= 3,
                         scoring= 'neg_mean_absolute_error',
                         return_train_score= True,
                         n_jobs= -1,
                         verbose= 1)

lasso_gcv_improved.fit(X_train_improved, y_train_improved)
```

In [82]:
```
lasso_model_improved= lasso_gcv_improved.best_estimator_
lasso_model_improved.fit(X_train_improved, y_train_improved)
```

Out[82]:
```
Lasso(alpha=0.0009, random_state=42)
```

In [83]:
```
lasso_coef_improved= pd.Series(lasso_model_improved.coef_, index= X_train_improved.columns)
selected_features_improved= len(lasso_coef_improved[lasso_coef_improved != 0])
print('Features selected by Lasso:', selected_features_improved)
```

```
Features selected by Lasso: 5
```

The optimal lamda=0.0009

features selected by lasso : 5

5.2 TRY TO SIMPLIFY THE MODEL

MSE:0.0460
R-squared: 0.705
adjusted R-squared: 0.690

In [208]:
```
lasso_coef_improved[lasso_coef_improved != 0]
```

Out[208]:
```
indus              0.003721
rm                 0.097068
ptratio           -0.000002
tax_log_trans     -0.189889
lstat_log_trans   -0.403924
dtype: float64
```

In [209]:
```
y_test_pred_improved= lasso_model_improved.predict(X_test_improved)
print( 'MSE on testing dataset:', mean_squared_error(y_test_improved, y_test_pred_improved))
```

```
MSE on testing dataset: 0.04602103658502262
```

In [213]:
```
print( 'r2 score on testing dataset:', r2_score(y_test_improved, y_test_pred_improved))
print('adj. r2 on testing dataset:',1 - (1 - r2_score(y_test_improved, y_test_pred_improved) ) * (1
```

```
r2 score on testing dataset: 0.7047735143600282
adj. r2 on testing dataset: 0.6893971348996131
```

Sanyal, S., Biswas, S. K., Das, D., Chakraborty, M., & Purkayastha, B. (2022, June). Boston house price prediction using regression models. *In 2022 2nd International Conference on Intelligent Technologies (CONIT)* (pp. 1-6). IEEE.

Model Name	R-Squared	RMSE	Cross-Validation
Simple Linear Regression	73.66%	4.329	73.17
Polynomial Regression (degree=2)	74.27%	4.279	73.17
Ridge Regression	88.28%	2.888	85.83
Lasso Regression	88.79%	2.833	85.57

06

CONCLUSION & FUTURE WORK

- All features except **‘indus’** was selected by Lasso regression as influential features for the house price in Boston. **12** out of 13 were selected by Lasso.
- The MSE of the model on testing set is around **0.0384**, its square root RMSE is around **0.1959**, which indicates that the model has a satisfying performance.
- Among multiple linear regression, polynomial regression, ridge regression and lasso regression, **polynomial regression** resulted in the best performance.
- Adjustments according the the reference article didn’t improve the performance of the Lasso model. Yet, a simpler Lasso model was derived.

- We didn’t manage to reproduce the good performance Lasso regression model in the article. However, we guess some further adjustments may lead us to the right path:

❏ Standarlize all the features and the house price;

❏ Try another encoding method on the categorical variables;

❏ Boost the sample size.

REFERENCES

1. Sanyal, S., Biswas, S. K., Das, D., Chakraborty, M., & Purkayastha, B. (2022, June). Boston house price prediction using regression models. *In 2022 2nd International Conference on Intelligent Technologies (CONIT)* (pp. 1-6). IEEE.
2. Zhao, R. (2020). Analysis of the Correlation between Housing Price Data in Boston Based on the Regression Method.
3. Xin, S. J., & Khalid, K. (2018). Modelling house price using ridge regression and lasso regression. *International Journal of Engineering & Technology*, 7(4.30), 498-501.
4. Wang, Y., & Zhao, Q. (2022, March). House Price Prediction Based on Machine Learning: A Case of King County. *In 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)* (pp. 1547-1555). Atlantis Press.

THANKS!