Carolina Carvalho Manhaes Leite (leite2)
IE598 MLF F19
Module 4 Homework (Regression)
09/22/2019

Part 1: Exploratory Data Analysis

Describe the data sufficiently using the methods and visualizations that we used previously in Module 3 and again this week. Include any output, graphs, tables, heatmaps, box plots, etc. Label your figures and axes. DO NOT INCLUDE CODE!

Split data into training and test sets. Use random_state = 42. Use 80% of the data for the training set. Use the same split for all models.

```
Assignment 4 - 09/22/2019
Name: Carolina Carvalho Manhaes Leite
NetID: leite2

------------

The complete dataset has 506 rows and 27 columns.

------------

Descriptive Statistics of Target Variable (MEDV)

count    452.000000
mean      23.750442
std        8.808602
min        6.300000
25%       18.500000
50%       21.950000
75%       26.600000
max       50.000000
Name: MEDV, dtype: float64

------------
```

The count of the target is less than the total number of rows. It means we have some missing values in the target value. While we could choose an input method to infer these values, to make it simple (and also because we'll lose just 10% of the original data), I'll drop them.

Now, performing some descriptive analysis (I'll display here the tables only for the original variables. The analysis for the noise variables can be found at the code in python.)

```
The dataset without the target missing values has 452 rows and 27 columns.
The features array has 452 rows and 26 columns.
The target array has 452 rows.
```

```
Descriptive Statistics of Original Variables

count    452.000000
mean       1.420825
std        2.495894
min        0.006320
25%        0.069875
50%        0.191030
75%        1.211460
max        9.966540
Name: CRIM, dtype: float64

count    452.000000
mean      12.721239
std       24.326032
min        0.000000
25%        0.000000
50%        0.000000
75%       20.000000
max      100.000000
Name: ZN, dtype: float64

count    452.000000
mean      10.304889
std        6.797103
min        0.460000
25%        4.930000
50%        8.140000
75%       18.100000
max       27.740000
Name: INDUS, dtype: float64

count    452.000000
mean       0.077434
std        0.267574
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
Name: CHAS, dtype: float64
```

```
count    452.000000
mean       0.540816
std        0.113816
min        0.385000
25%        0.447000
50%        0.519000
75%        0.605000
max        0.871000
Name: NOX, dtype: float64

count    452.000000
mean       6.343538
std        0.666808
min        3.561000
25%        5.926750
50%        6.229000
75%        6.635000
max        8.780000
Name: RM, dtype: float64


count    452.000000
mean      65.557965
std       28.127025
min        2.900000
25%       40.950000
50%       71.800000
75%       91.625000
max      100.000000
Name: AGE, dtype: float64

count    452.000000
mean       4.043570
std        2.090492
min        1.129600
25%        2.354750
50%        3.550400
75%        5.401100
max       12.126500
Name: DIS, dtype: float64

count    452.000000
mean       7.823009
std        7.543494
min        1.000000
25%        4.000000
50%        5.000000
75%        7.000000
max       24.000000
Name: RAD, dtype: float64
```

```
count    452.000000
mean     377.442478
std      151.327573
min      187.000000
25%      276.750000
50%      307.000000
75%      411.000000
max      711.000000
Name: TAX, dtype: float64

count    452.000000
mean      18.247124
std        2.200064
min       12.600000
25%       16.800000
50%       18.600000
75%       20.200000
max       22.000000
Name: PTRATIO, dtype: float64

count    452.000000
mean     369.826504
std       68.554439
min        0.320000
25%      377.717500
50%      392.080000
75%      396.157500
max      396.900000
Name: B, dtype: float64


count    452.000000
mean      11.441881
std        6.156437
min        1.730000
25%        6.587500
50%       10.250000
75%       15.105000
max       34.410000
Name: LSTAT, dtype: float64

------------
```

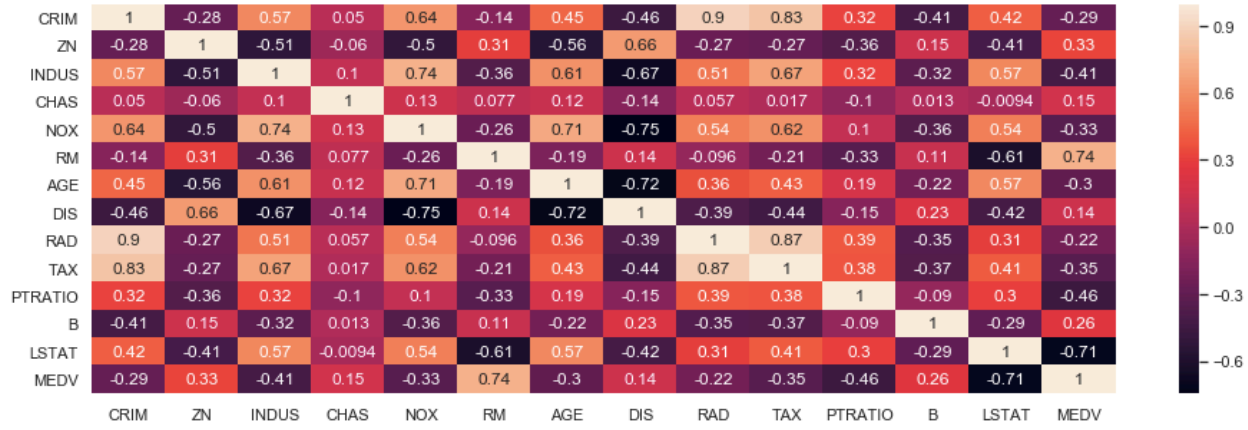
```
Descriptive Statistics of Target Variable (MEDV)

count    452.000000
mean      23.750442
std        8.808602
min        6.300000
25%       18.500000
50%       21.950000
75%       26.600000
max       50.000000
Name: MEDV, dtype: float64

------------
```

Taking a look at the correlations' heat map:

Heat Map of Original Variables + Target

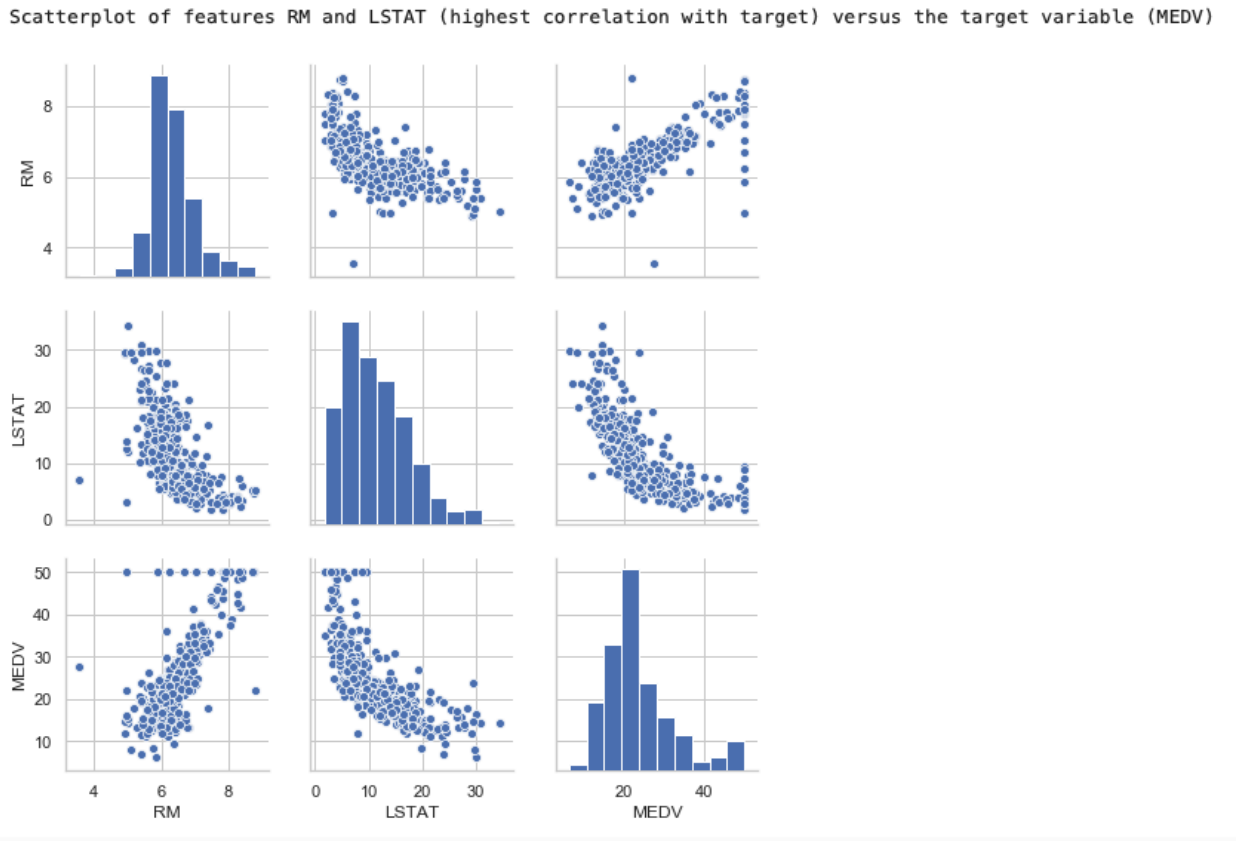


We can reach some conclusions based on the matrix above:

- Some features are highly correlated among themselves (for example, "RAD"/"CRIM", with a correlation of 0.9, and "TAX"/"CRIM", with a correlation of 0.83). This could represent a criterion for us to eliminate some variables. We have a small dataset, though (only 13 variables – without mentioning the "noises"), so instead of performing a feature selection based on this metric, we'll compute the correlation between all features and the target.
- All features are somewhat correlated to the target variable (there is no correlation below 0.1);
- Two features stand out as highly correlated with the target: "RM" (positively correlated; 0.74) and "LSTAT" (negatively correlated; -0.71).

We'll let the models "decide" which features should stay. The linear regression won't eliminate any variable, but we'll be able to see the least important ones by their low coefficient. LASSO, on the other hand, it's widely used for feature selection process, so we might expect some coefficients equal to 0. Another expected behavior is that both variables mentioned above as highly correlated with the target ("RM" and "LSTAT") should appear as the strongest variables in our models.

Just to give us a feeling of the actual relationship between the two most correlated variables and the target, I build a scatter plot between them:



We can confirm the positive correlation between “RM” and “MEDV” (when one gets higher the other gets too) and the negative correlation between “LSTAT” and “MEDV”. More than that, we can infer that the relation between “RM” and “MEDV” is linear, while the relation “LSTAT” x “MEDV” is clearly not linear. Another insight is that all distributions don’t seem to be *that* different from a normal one, which is a good thing, since most of the methods we’ll apply assume normal distributions. Just in case (because we haven’t plotted the other variables), we performed a standardization in all variables.

Part 2: Linear regression

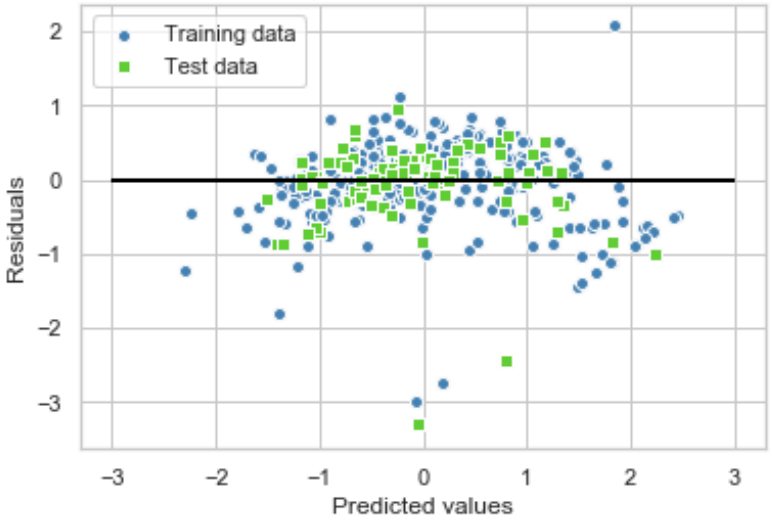
Fit a linear model using SKlearn to all of the features of the dataset. Describe the model (coefficients and y intercept), plot the residual errors, calculate performance metrics: MSE and R2.

We fitted a linear model to all of the features. As a result:

```
Coefficients:
[[ 7.15110939e-02 -9.88995606e-03  2.80872587e-02 -3.90217733e-04
  -9.49853873e-03 -2.63197715e-02 -3.79754117e-03 -2.50782070e-02
  -1.49881400e-02 -2.82417393e-02 -4.23027274e-02  3.75734312e-02
  -5.30381150e-03 -5.37158693e-02  8.61306834e-02  3.86234100e-02
   5.56777624e-02 -1.59637634e-01  4.22835484e-01 -7.23447518e-02
  -3.25138222e-01  2.13878961e-01 -1.67595810e-01 -2.15974719e-01
   1.04777233e-01 -3.23565661e-01]]

Intercept: 0.000

Residual Plot
```




```
Metrics

MSE train: 0.241
MSE test: 0.325
R2 train: 0.759
R2 test: 0.675
```

The features are sorted in the following order: ATT1-13 (noise variables), CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B and LSTAT, and that's exactly the order in which the coefficients are displayed above. We can see that most of our noise variables have low absolute coefficients (one of them even around 10^{-4} . The original features have higher absolute coefficients (indicating that they contribute more to explain the variance of the data). However, we can see some unexpected results, with some of the noise variables showing higher absolute coefficients than the original one (ATT1's absolute coefficient, for example, is greater than CRIM's). This may lead to surprising results when fitting the data with Ridge and LASSO Regressions. I'll talk about that later.

The fact that the intercept is zero is a direct consequence of the variables' standardization.

Part 3.1: Ridge regression

Fit a Ridge model using SKlearn to all of the features of the dataset. Test some settings for alpha. Describe the model (coefficients and y intercept), plot the residual errors, calculate performance metrics: MSE and R2. Which alpha gives the best performing model?

I've changed the alpha parameter in order to draw some conclusions about its influence in the model performance. Let's see some of the results (the more interesting/representative ones. I tried several alpha values, and the details can be seen at the code).

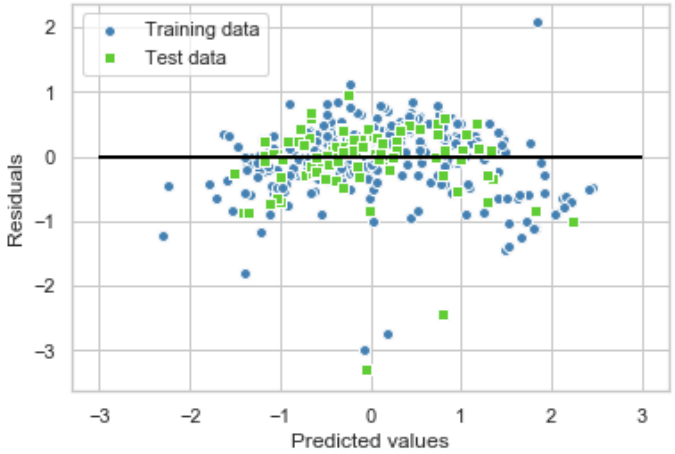
- **Alpha = 0**

```
Alpha = 0.0

Coefficients:
[[ 7.15110939e-02 -9.88995606e-03  2.80872587e-02 -3.90217733e-04
  -9.49853873e-03 -2.63197715e-02 -3.79754117e-03 -2.50782070e-02
  -1.49881400e-02 -2.82417393e-02 -4.23027274e-02  3.75734312e-02
  -5.30381150e-03 -5.37158693e-02  8.61306834e-02  3.86234100e-02
   5.56777624e-02 -1.59637634e-01  4.22835484e-01 -7.23447518e-02
  -3.25138222e-01  2.13878961e-01 -1.67595810e-01 -2.15974719e-01
   1.04777233e-01 -3.23565661e-01]]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.241
MSE test: 0.325
R2 train: 0.759
R2 test: 0.675
```

As expected, the results are exactly the same as the linear regression without penalization.

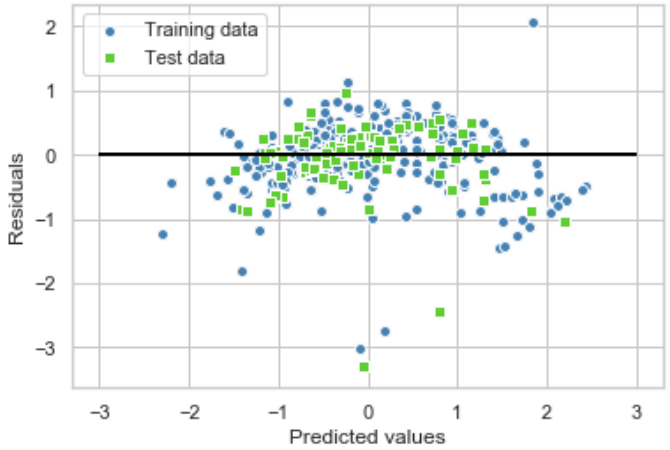
- **Alpha = 0.01**

```
Alpha = 0.01

Coefficients:
[[ 0.07030926 -0.01082436  0.02740597 -0.00160899 -0.00979384 -0.02560383
  -0.00265375 -0.02485246 -0.01467851 -0.02769861 -0.04167714  0.0370928
  -0.0055469  -0.04367441  0.08045989  0.02938611  0.05698962 -0.15062086
   0.42115769 -0.07010354 -0.31320683  0.18892466 -0.15072596 -0.21304492
   0.10488608 -0.32363662]]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.241
MSE test: 0.326
R2 train: 0.759
R2 test: 0.674
```

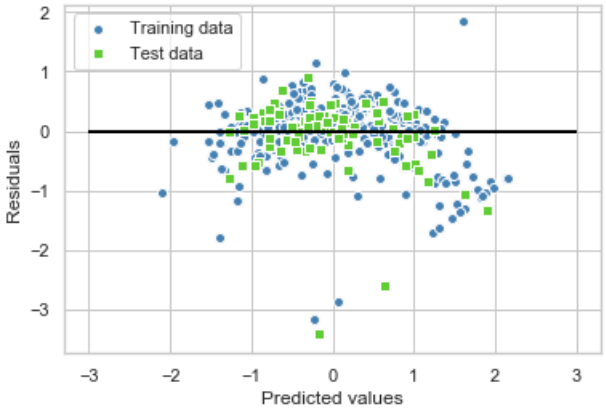
- **Alpha = 0.2**

```
Alpha = 0.2

Coefficients:
[[ 0.05346777 -0.01857146  0.0232585  -0.01061868 -0.00931658 -0.02017649
   0.00365616 -0.01963214 -0.01106731 -0.02359436 -0.03336591  0.03184065
  -0.00501866 -0.01396713  0.0464471  -0.0273299   0.06364892 -0.07279373
   0.37937009 -0.04353088 -0.19212614  0.06536641 -0.06800829 -0.18375737
   0.0946172  -0.29847339]]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.257
MSE test: 0.340
R2 train: 0.743
R2 test: 0.660
```

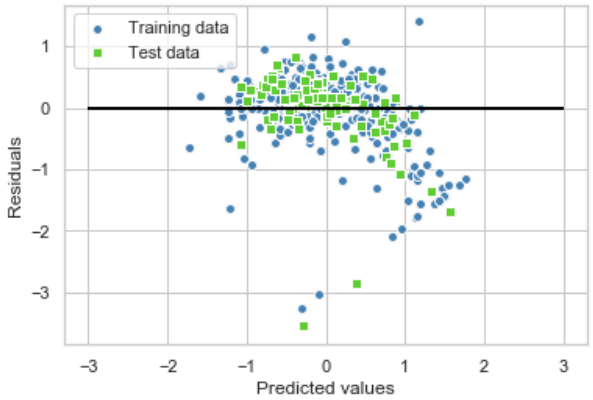
- **Alpha = 0.7**

```
Alpha = 0.7

Coefficients:
[[ 0.03339349 -0.02052258  0.02014782 -0.01545731 -0.00612653 -0.01492081
   0.00593587 -0.01218874 -0.00667715 -0.01948526 -0.02185943  0.0240303
  -0.00185609 -0.01838326  0.03728326 -0.04792988  0.05914774 -0.03896116
   0.29383347 -0.02860341 -0.09791996  0.02047092 -0.04770542 -0.14674907
   0.07307689 -0.2345941 ]]

Intercept: 0.000

Residual Plot
```

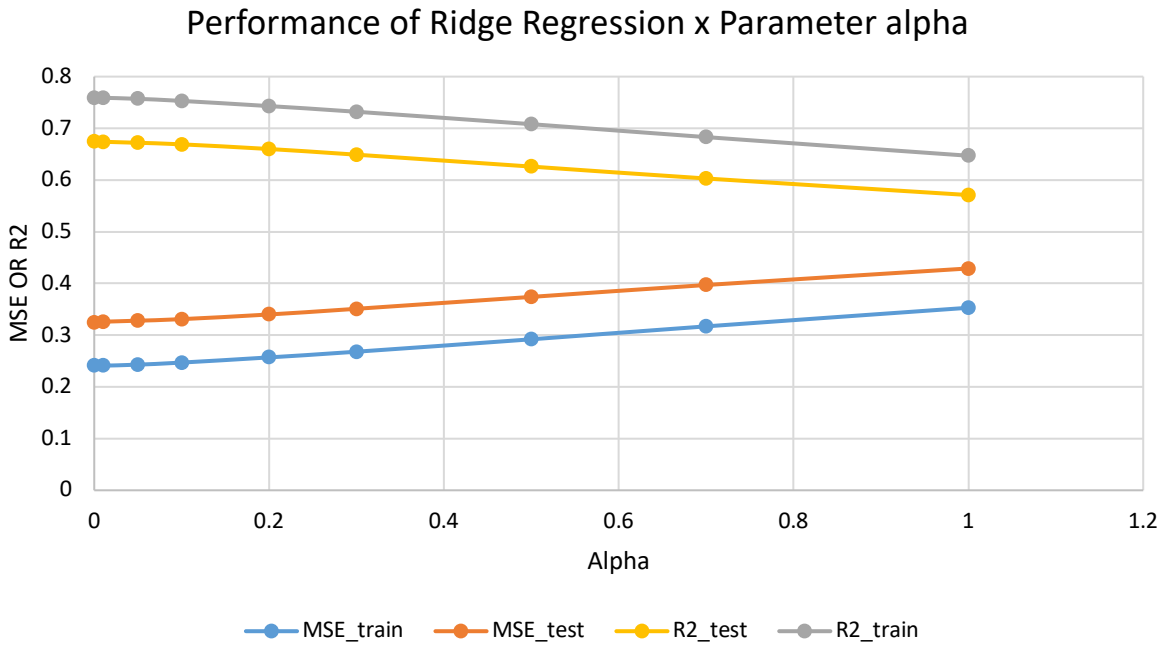



```
MSE train: 0.317
MSE test: 0.397
R2 train: 0.683
R2 test: 0.603
```

Finally, putting together all the alphas and their respective MSE_train, MSE_test, R2_train and R2_test:

Alpha	MSE_train	MSE_test	R2_train	R2_test
0	0.241	0.325	0.759	0.675
0.01	0.241	0.326	0.759	0.674
0.05	0.243	0.328	0.757	0.672
0.1	0.247	0.331	0.753	0.669
0.2	0.257	0.340	0.743	0.660
0.3	0.268	0.351	0.732	0.649
0.5	0.292	0.374	0.708	0.626
0.7	0.317	0.397	0.683	0.603
1	0.353	0.429	0.647	0.571

Graphically:



We can see from both the table and the chart that, for this specific case, the Ridge Regression performs less as alpha gets higher (MSE gets higher and, consequently, R2 gets lower), with the best alpha (not equal zero) being 0.01. We can see that the performance parameters for alpha = 0.01 are very close to those for alpha = 0 (linear regression without penalization). It may be possible that an inflexion point exists, and if it does, the data suggests it's located between 0 and 0.01 (i.e., a low alpha). We'll discuss more this aspect in the Conclusion section of this report.

Part 3.2: LASSO regression

Fit a LASSO model using SKlearn to all of the features of the dataset. Test some settings for alpha. Describe the model (coefficients and y intercept), plot the residual errors, calculate performance metrics: MSE and R2. Which alpha gives the best performing model?

I've changed the alpha parameter the same way I did in the Ridge Regression. Let's see some of the results:

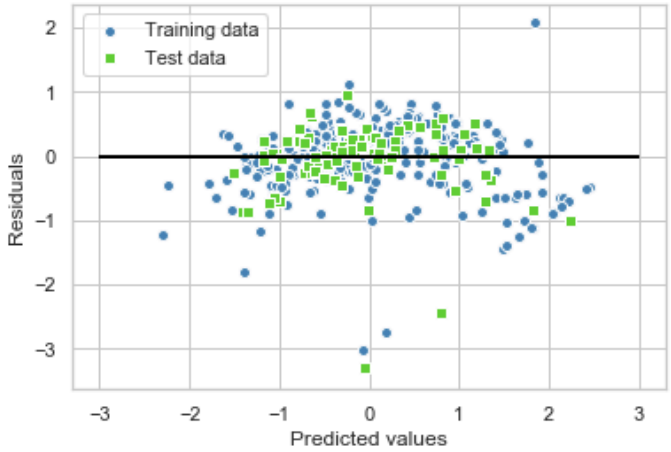
- **Alpha = 0.00001**

```
Alpha = 1e-05

Coefficients:
[ 7.13073634e-02 -9.74151098e-03  2.78719198e-02 -3.04593743e-04
 -9.36669789e-03 -2.60641099e-02 -3.34161400e-03 -2.48622286e-02
 -1.46927643e-02 -2.80234812e-02 -4.21134964e-02  3.73739886e-02
 -5.25193788e-03 -5.17563502e-02  8.52718039e-02  3.69467268e-02
  5.56830022e-02 -1.58817127e-01  4.22769910e-01 -7.18325375e-02
 -3.24101384e-01  2.10243858e-01 -1.65219151e-01 -2.15641953e-01
  1.04738223e-01 -3.23909505e-01]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.241
MSE test: 0.325
R2 train: 0.759
R2 test: 0.675
```

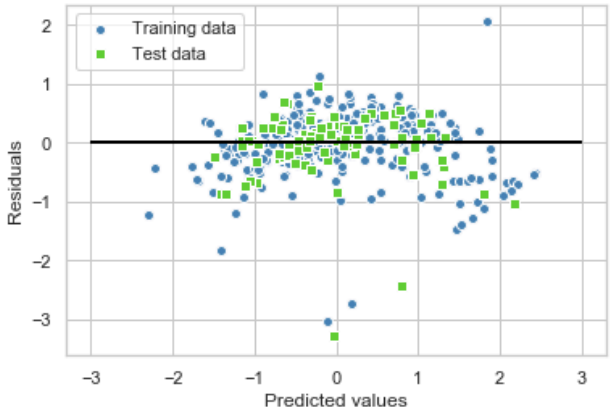
- **Alpha = 0.0001**

```
Alpha = 0.0001

Coefficients:
[ 0.06949056 -0.00845337  0.02595204 -0.         -0.00820279 -0.02375434
 -0.         -0.02297226 -0.0120925  -0.02605261 -0.0404281   0.03551549
 -0.00476775 -0.03351864  0.07736074  0.02183161  0.05575676 -0.15118056
  0.42231074 -0.06746084 -0.3145565   0.17654087 -0.1434191  -0.21259296
  0.10448337 -0.32700402]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.241
MSE test: 0.326
R2 train: 0.759
R2 test: 0.674
```

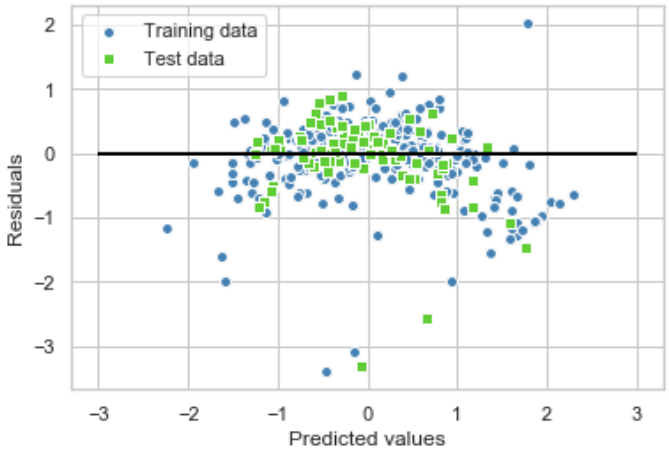
- **Alpha = 0.0025**

```
Alpha = 0.0025

Coefficients:
[ 0.0126011  -0.          0.         -0.         -0.         -0.
  0.         -0.         -0.         -0.         -0.          0.
 -0.         -0.          0.         -0.          0.02453367 -0.
  0.42737293 -0.         -0.06652127 -0.         -0.00130912 -0.17743243
  0.07456798 -0.34898286]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.282
MSE test: 0.356
R2 train: 0.718
R2 test: 0.644
```

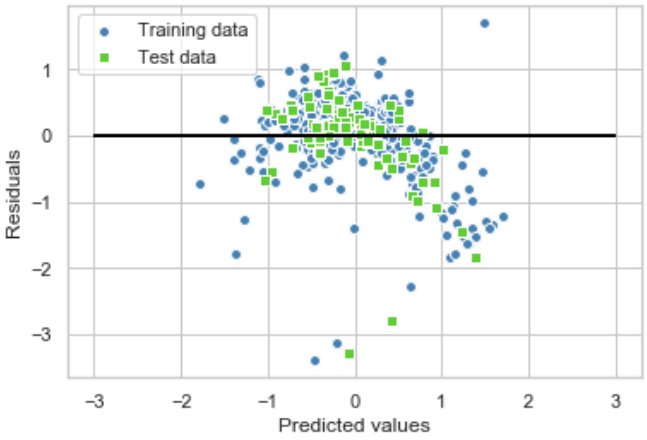
- **Alpha = 0.01**

```
Alpha = 0.01

Coefficients:
[ 0.         -0.          0.         -0.         -0.         -0.
  0.         -0.          0.         -0.         -0.          0.
 -0.         -0.          0.         -0.          0.         -0.
  0.36911492 -0.         -0.         -0.         -0.         -0.08036357
  0.         -0.26556552]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.365
MSE test: 0.437
R2 train: 0.635
R2 test: 0.563
```

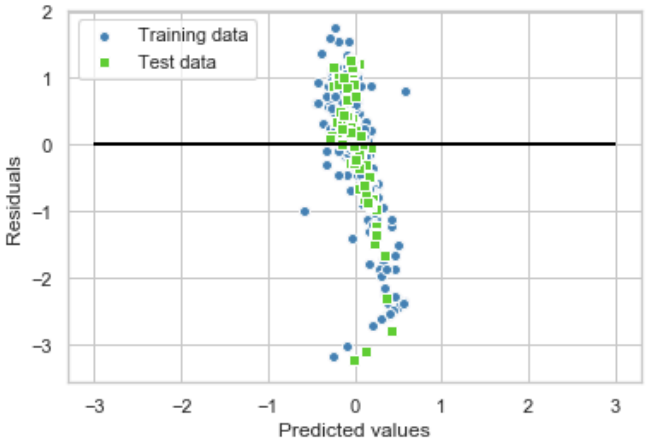
- **Alpha = 0.03**

```
Alpha = 0.03

Coefficients:
[ 0.          -0.           0.          -0.          -0.          -0.
  0.          -0.           0.          -0.          -0.           0.
 -0.          -0.           0.          -0.           0.          -0.
  0.15199686 -0.           0.          -0.          -0.          -0.
  0.          -0.03968598]

Intercept: 0.000

Residual Plot
```

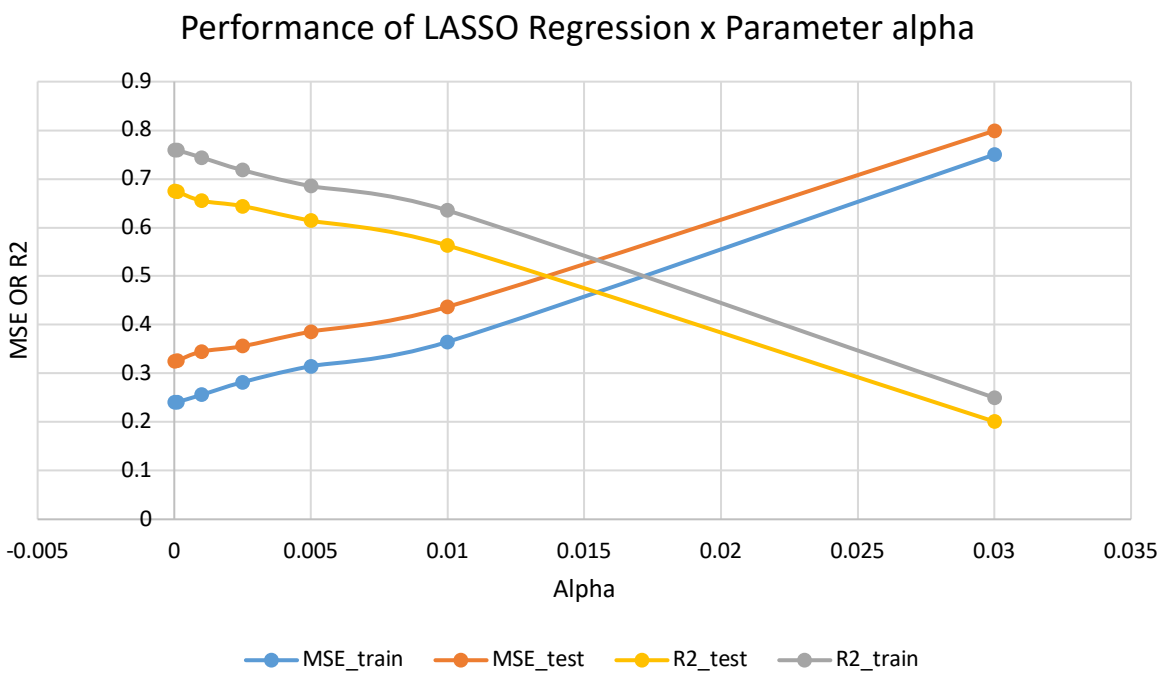



```
MSE train: 0.750
MSE test: 0.799
R2 train: 0.250
R2 test: 0.201
```

Even before summarizing the data, we can already draw some conclusions. Besides the fact that there was no convergence for alpha = 0, we can see a trend in the coefficients movements'' with higher values of alpha: each time more and more coefficients were set to zero (with might happen when using LASSO), up until the point that only two variables would be relevant to the model: the exact same features with the highest correlations with the target variable. Another interesting aspect of LASSO is the magnitude of the alpha values: they're much smaller than the alphas used for the Ridge Regression.

Summarizing the LASSO results:

Alpha	MSE_train	MSE_test	R2_train	R2_test
0.00001	0.241	0.325	0.759	0.675
0.0001	0.241	0.326	0.759	0.674
0.001	0.256	0.345	0.744	0.655
0.0025	0.282	0.356	0.718	0.644
0.005	0.315	0.386	0.685	0.614
0.01	0.365	0.437	0.635	0.563
0.03	0.750	0.799	0.250	0.201



Again, we can see from the chart that, for this specific case, the LASSO Regression performs less as alpha gets higher (MSE gets higher and, consequently, R2 gets lower), with the best alpha being 0.00001. The performance parameters for alpha = 0.00001 are exactly the same as the ones for the linear regression without penalization. Again, an Inflexion point may exist, and if it does, the data suggests it's located between 0 and 0.00001 (i.e., a low alpha).

Part 4: Conclusions

Write a short paragraph summarizing your findings.

Comparing the three models (linear regression without penalization, Ridge regression and LASSO regression), we see that the option with highest performance was the regression without penalization. We were even able to find very close (or even equal) results for the other models, but always with low values of alpha. There're two aspects to be considered here: the first one is that we didn't perform a K-fold cross validation process, so it's possible that the test results (obtained from just one sample) may be somewhat biased – in the case of the randomly chosen test sample doesn't represent every behavior of the population as a whole. Another interesting highlight is the influence of the 13 noise features in the dataset. It's also possible that some of these randomly generated variables have, by chance, some correlation with the target, so that the model loses performance when they're not being considered. Just out of curiosity, I performed to additional tests. The first one is the Pearson correlation between each of the noise features and the target, and then I build all of the three models again, without these variables:

```
Correlation of Noise Variables (ATT) with the Target Variable (MEDV)

ATT1: 0.03527790774788591
ATT2: -0.05524994108680068
ATT3: 0.08621956498444258
ATT4: -0.05870079458257703
ATT5: -0.03704323879945478
ATT6: -0.03578272022590616
ATT7: 0.044602840583505096
ATT8: 0.0011677655149674635
ATT9: 0.016676734001531548
ATT10: -0.08121443854713163
ATT11: -0.0005179534326979884
ATT12: 0.04636772540478101
ATT13: -0.048623991014759865
```

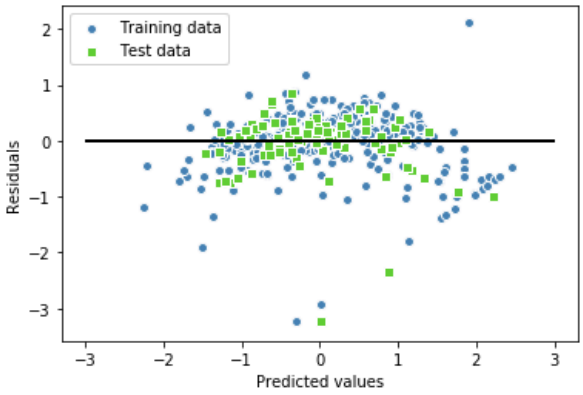
All absolute correlations are low, but they're not essentially equal to 0.

- **Linear Regression without Penalization**

```
Coefficients:
[[-0.05613477  0.07660477  0.04027393  0.0562009  -0.15176139  0.41473622
  -0.06428269 -0.30781607  0.20913074 -0.15722199 -0.22034414  0.10519591
  -0.33791816]]

Intercept: 0.000

Residual Plot
```




```
Metrics

MSE train: 0.253
MSE test: 0.309
R2 train: 0.747
R2 test: 0.691
```

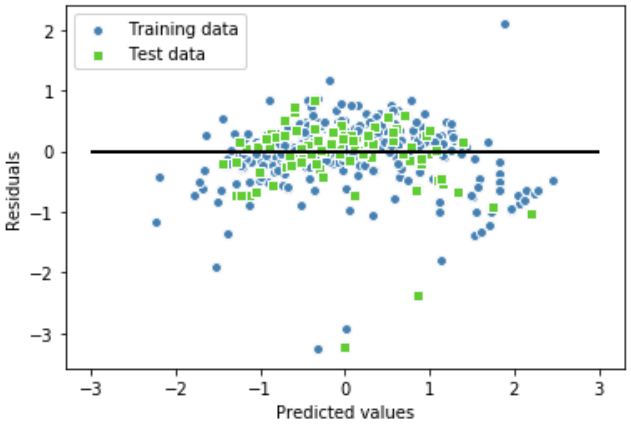
- **Ridge Regression (alpha = 0.01)**

```
Alpha = 0.01

Coefficients:
[[-0.04682988  0.07158558  0.03152538  0.05740754 -0.14331409  0.41370996
  -0.06288135 -0.29716458  0.18594709 -0.14169335 -0.21727251  0.10526156
  -0.33700571]]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.253
MSE test: 0.311
R2 train: 0.747
R2 test: 0.689
```

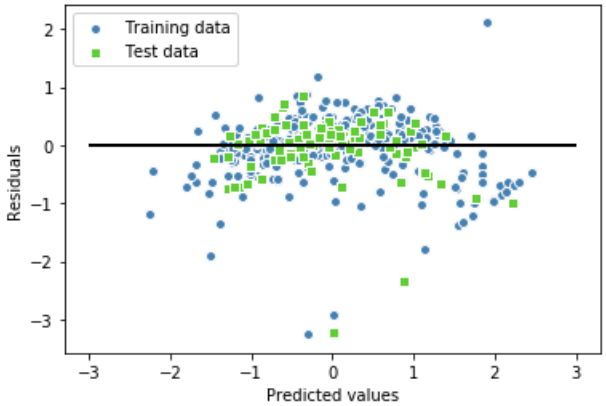
- **LASSO Regression (alpha = 0.00001)**

```
Alpha = 1e-05

Coefficients:
[-0.05432604  0.07582213  0.03870035  0.05619858 -0.1509012   0.41475458
 -0.06393133 -0.30685519  0.20568253 -0.15498787 -0.21998587  0.10515594
 -0.33812183]

Intercept: 0.000

Residual Plot
```




```
MSE train: 0.253
MSE test: 0.309
R2 train: 0.747
R2 test: 0.691
```

From the new regressions (without the noise variables), only the model with no penalization showed a better performance (MSE_test = 0.309 vs 0.325 for the previous model). We still haven't detected any significative change when using Ridge or LASSO. It's still possible that the presence of only one test sample represent a bias to the analysis, which could be confirmed after a K-fold cross validation process.

Part 5: Appendix

GitHub link: **https://github.com/leiteccml/Carolina_2019**

Part 6: DataCamp Assigments