**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.**

----

A screenshot of a cell phone

Description automatically generated

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.)

A screenshot of a cell phone

Description automatically generated

A screenshot of text

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Taking a look at the correlations’ heat map:

A screenshot of a cell phone

Description automatically generated

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:

A screenshot of a cell phone

Description automatically generated

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:

A close up of text on a white background

Description automatically generated

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 . 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**

A screenshot of a cell phone

Description automatically generated

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

* **Alpha = 0.01**

**A close up of a map

Description automatically generated**

* **Alpha = 0.2**

**A close up of text on a white background

Description automatically generated**

* **Alpha = 0.7**

A screenshot of text

Description automatically generated

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**

A close up of text on a white background

Description automatically generated

* **Alpha = 0.0001**

**A close up of text on a white background

Description automatically generated**

* **Alpha = 0.0025**

**A close up of a map

Description automatically generated**

* **Alpha = 0.01**

**A close up of a map

Description automatically generated**

* **Alpha = 0.03**

**A close up of a map

Description automatically generated**

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:

A screenshot of a cell phone

Description automatically generated

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

* **Linear Regression without Penalization**

A screenshot of a cell phone

Description automatically generated

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

**A screenshot of a cell phone

Description automatically generated**

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

A screenshot of a cell phone

Description automatically generated

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**

**A screenshot of a cell phone

Description automatically generated**