DHRUV GARG 16 BCE 1190

MACHINE LEARNING - DIGITAL ASSIGNMENT 1

REGULARIZATION WITH LINEAR REGRESSION

INTRODUCTION

Before getting started with the implementation, a few terms to be understood:

Overfitting: Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data (test data). The model will have a low accuracy if it is overfitting.

Regularization: This form of regression, constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting. The fitting procedure involves a loss function, known as residual sum of squares or RSS.

$$\mathrm{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$$

Ridge regression (L2 regularization): RSS is modified by adding the shrinkage quantity. λ is the tuning parameter that decides how much we want to penalize the flexibility of our model. When λ = 0, the penalty term has no effect, and the estimates produced by ridge regression will be equal to least squares. However, as λ→∞, the impact of the shrinkage penalty grows, and the ridge regression coefficient estimates will approach zero. Ridge regression uses squares of β, as its penalty.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \mathrm{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$$

Lasso regression (L1 regularization): Lasso is another variation, in which the above function is minimized. Its clear that this variation differs from ridge regression only in penalizing the high coefficients. It uses |βj|(modulus)instead of squares of β, as its penalty.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \mathrm{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|.$$

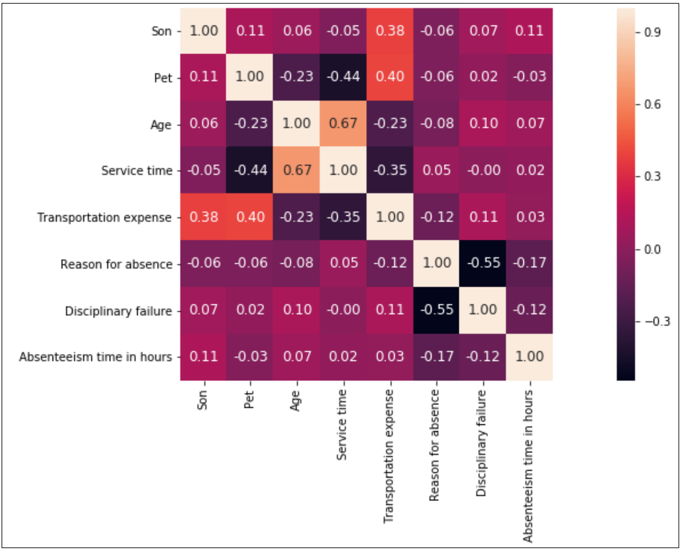
Elastic net regression ( ): Elastic net regression overcomes the limitations of both methods – Ridge and Lasso. The formula for Elastic net in terms of the above two equations is:

$$\text{Residual Mean Square Error} + \alpha\cdot\text{Ridge Penalty} + (1-\alpha)\cdot\text{LASSO Penalty}$$

$$\alpha \in [0,1].$$

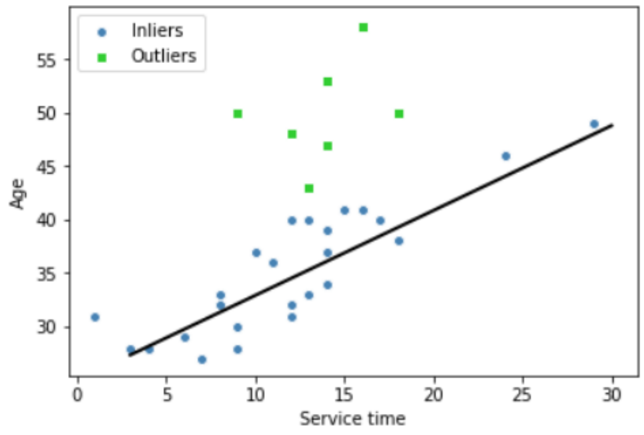
IMPLEMENTATION

First the data was read, pre-processed and the dataset values were scaled using the StandardScaler. Scaling is very important so as to prevent one/more attributes from being given undue importance with respect to the others. Next, two attributes were identified from the dataset that were linearly dependent to each other, i.e. had a high correlation coefficient. To arrive at these attributes, multiple correlation matrices were plot. Thus two attributes having a good correlation coefficient (0.67) were chosen.



Linear regression using Sklearn's LinearRegression class

The values of the two identified attributes were stored into arrays – X and y. The Ransack Regressor was used to perform linear regression. It does a very good job since the Ransack Regressor identifies and eliminates the outliers from the data while fitting the line. [For the Ransack regression, unscaled data was used]

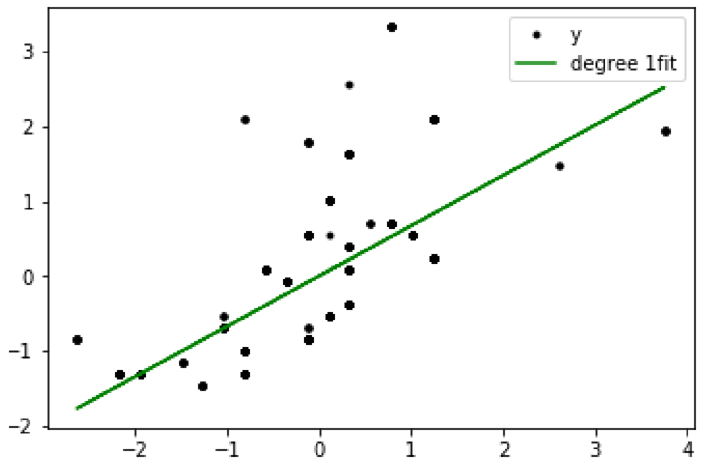


UNDERSTANDING REGULARIZATION THROUGH IMPLEMENTATION

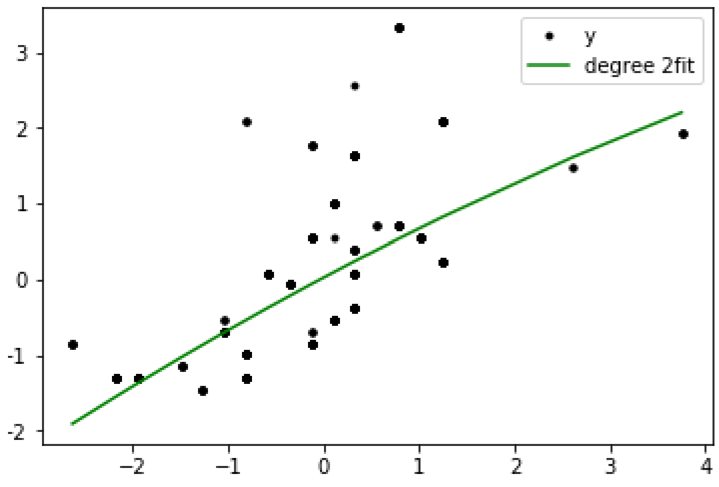
A function "get_lm" was defined to obtain the linear regression model for various degrees of polynomial. This was the function to which data (X,y) was passed and various polynomial regression models were obtained. We obtained different polynomial model curves for degrees 1, 2, 3, 4, 10, 15, and 20.

We observe that as the degree increases, the tendency for overfitting increases. At degree = 20, the training data was completely memorized by the algorithm.

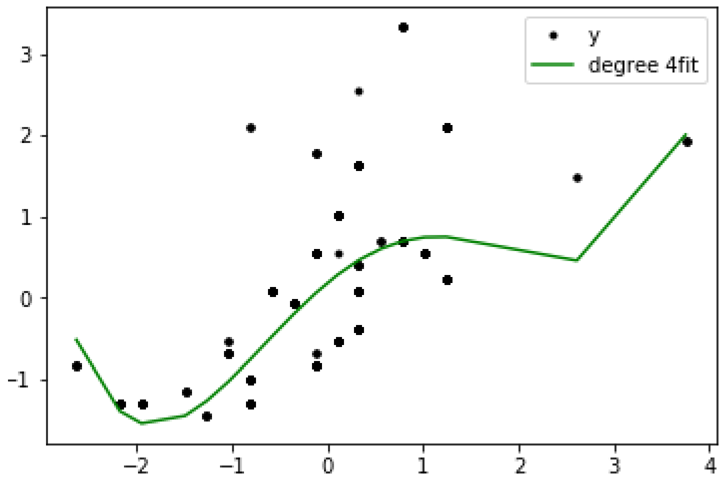
Degree 1



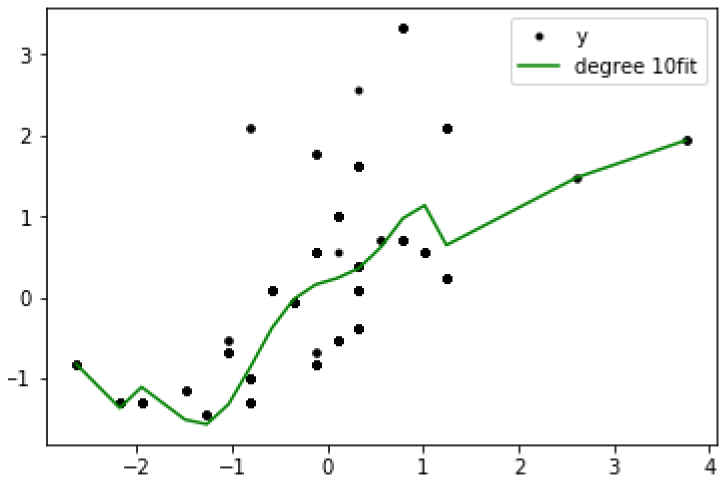
Degree 2



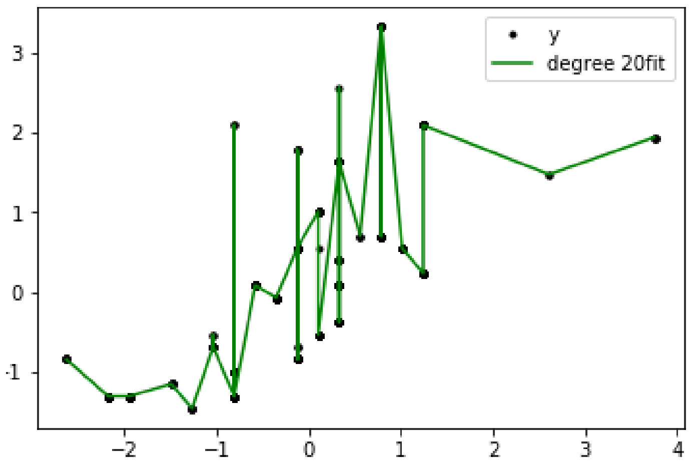
Degree 4



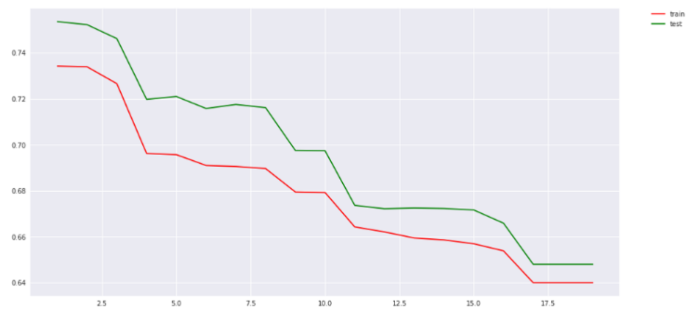
Degree 10



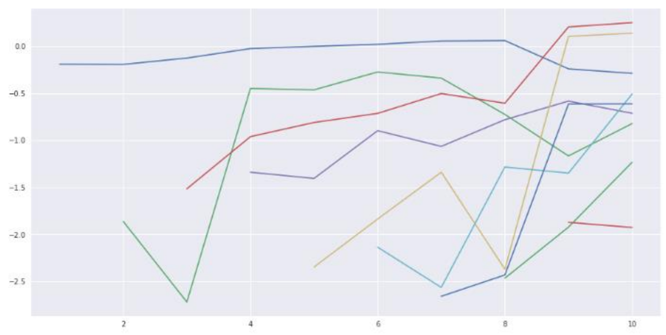
Degree 20



To visualize the models of various degrees – along with training sample error and out of training sample error, a for loop was run from 1 to 20, and a dataframe was created - storing all the coefficients of the polynomials, along with **train and test mean-squared errors**.



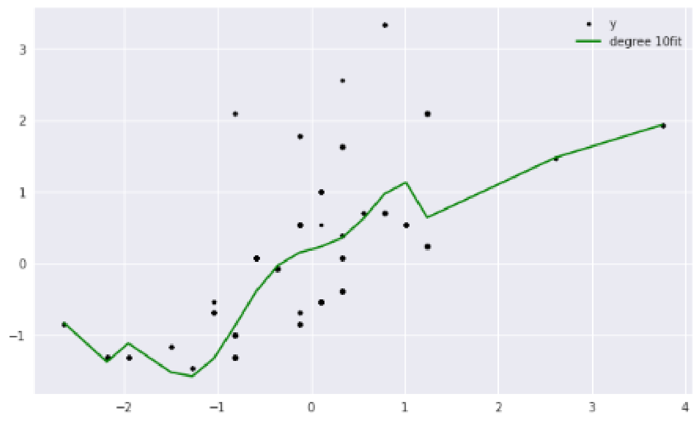
We also observe how the parameter coefficients changes as the models become more complex. As seen from the image shown, the coefficients have a very haphazard pattern, but overall following an upward trend, ie. as the **complexity of the model increases, the value of the coefficients also increases.**



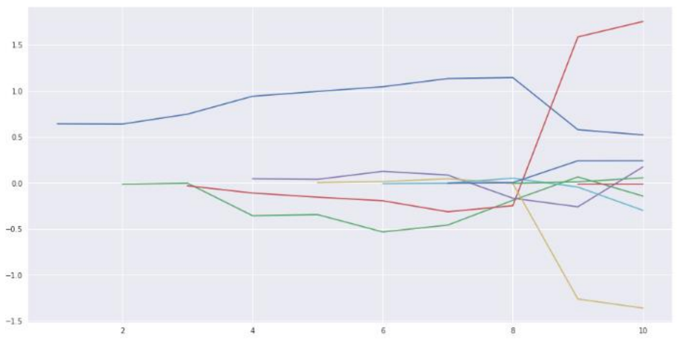
RIDGE REGULARIZATION

We import Ridge class from sklearn.linear_model and obtain the regularized polynomial curve for degree = 10. The plot is shown below.

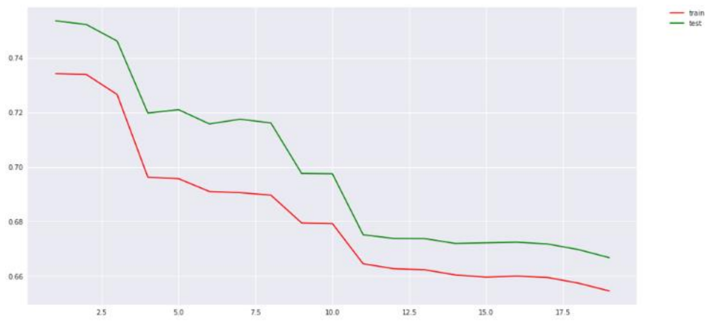
```
Ridge(alpha=0.01, copy_X=True, fit_intercept=True, max_iter=None,
   normalize=False, random_state=None, solver='auto', tol=0.001)
```



Further a loop from 1 to 20 is used to obtain the coefficient values of all models formed using ridge regularization. The **variation in coefficients is much lesser** when compared to original regression (without regualrization).

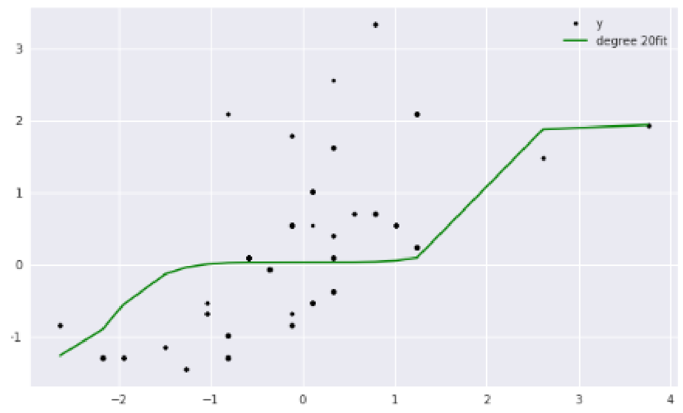


Finally, from the obtained models, the training data error and testing data error are plotted. <u>The final squared error found was lowest for model with degree 19. It gave a squared error of 0.64. A similar pattern was observed in the original plot (without regularization) and it gave an out-of-sample squared error of 0.65.</u>

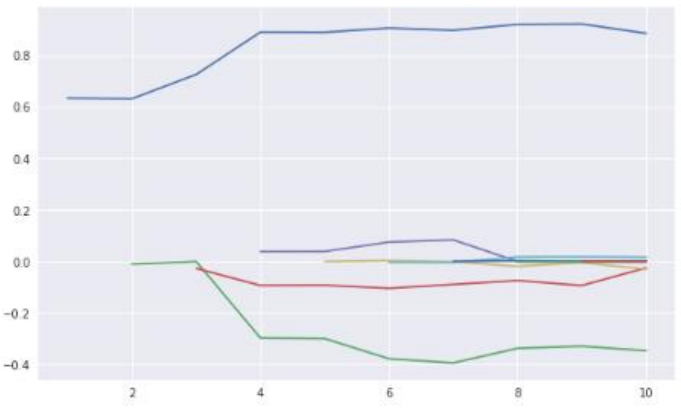


LASSO REGULARIZATION

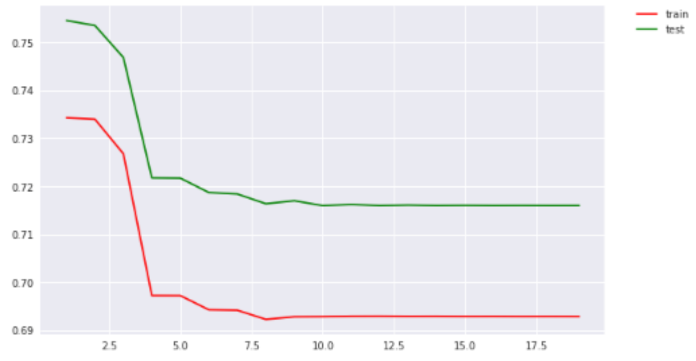
We import Lasso class from sklearn.linear_model and obtain the regularized polynomial curve for degree = 20. The plot is shown below. **As we can see, it did not memorize the data for degree=20 as had happened without regularization.**



Further a loop from 1 to 20 is used to obtain the coefficient values of all models formed using lasso regularization. The **variation in coefficients is much lesser** when compared to original regression (without regualrization).



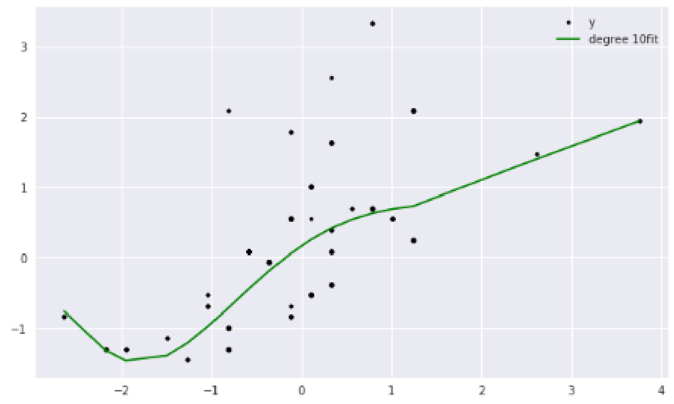
Finally, from the obtained models, the training data error and testing data error are plotted. <u>The final squared error found was lowest for model with degree 8. It gave a squared error of 0.69. A much different pattern was observed in the original plot (without regularization) and it gave an out-of-sample squared error of 0.65. Here we see that there is an initial fall in the mean-squared-error but after degree=8, the error increases slightly and is almost constance henceforth.</u>



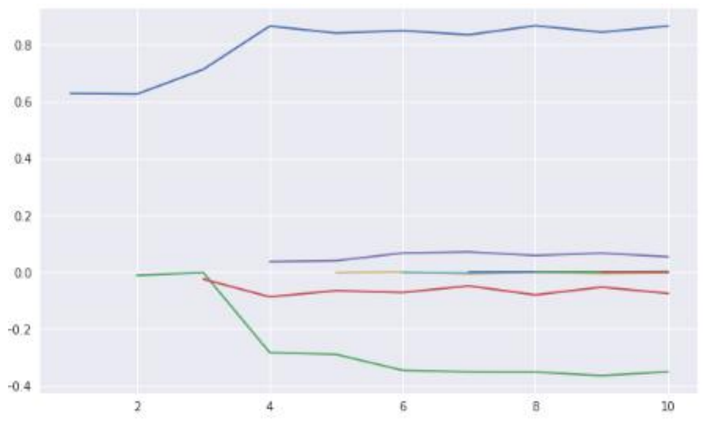
ELASTIC NET REGULARIZATION

We import ElasticNet class from sklearn.linear_model and obtain the regularized polynomial curve for degree = 10. The plot is shown below. **As we can see, it is somewhat similar (but smoother) than the model without regularization.**

```
ElasticNet(alpha=0.02, copy_X=True, fit_intercept=True, l1_ratio=0.4,
      max_iter=1000, normalize=False, positive=False, precompute=False,
      random_state=None, selection='cyclic', tol=0.0001, warm_start=False)
```

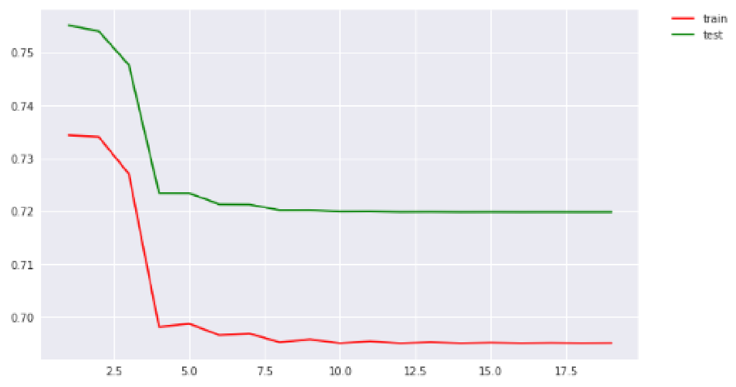


Further a loop from 1 to 20 is used to obtain the coefficient values of all models formed using lasso regularization. The **variation in coefficients is much lesser** when compared to original regression (without regualrization).



Finally, from the obtained models, the training data error and testing data error are plotted. <u>The final squared error found was lowest for model with degree 19. It gave a squared error of 0.69. A much different pattern was observed in the original plot</u>

(without regularization) and it gave an out-of-sample squared error of 0.65. Here we see that there is an initial fall in the mean-squared-error but after degree=10, the error decreases gradually henceforth.



RESULT:

As we could observe, the overfitting of data is eliminated by using regularization. This is not only seen through the test-sample-error vs training-sample-error plots, we can also observe that the coefficients of the model show lesser variation. This can be attributed to regularization – which restricts the variation in the coefficients.