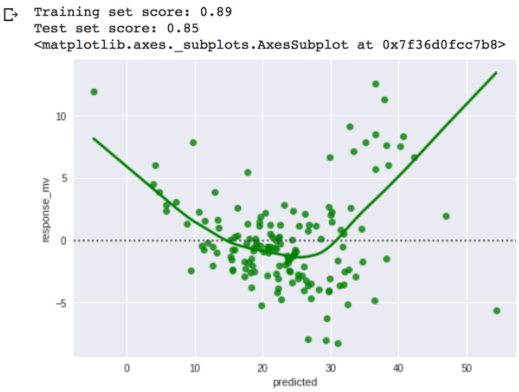
Introduction: Summary and problem definition for management

Assignment 4 builds on the linear regression models we developed for assignment 3. The data set in this assignment is the Boston housing data from a book about regression diagnostics by Belsley, Kuh, and Welsch.[1] The machine learning methods used are a reflection of the linear regression chapters in our course textbook.[2] In the prior analysis, we investigated 3 modeling techniques that applied a standard linear regression, a Ridge model and a Lasso model. In the end, the recommendation was to apply a Lasso technique with slight constraints on the degrees of freedom to avoid over-fitting.

The resulting residual plot from assignment 3 is shown here on the right. Although the training and test scores fall within an acceptable range, the curved arc of the residual plot suggests that we may need to work on the fit a bit more. In this week's assignment, I experiment with decision trees and random forest techniques to improve on the model fit.

Assignment 3 – 'Best Fit' Model using Lasso Technique

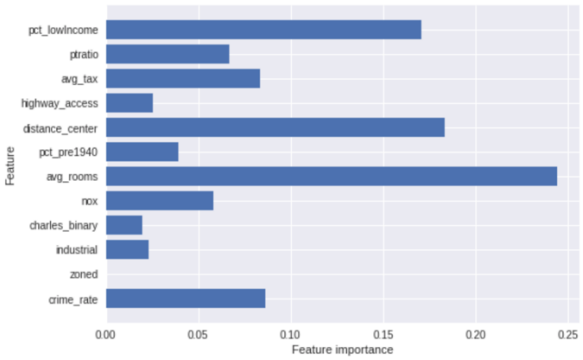


**suggested reference point for Assignment 4*

Research Design, Measurement & Statistical Methods

As the data set is the same in assignment 3, there were no additional discoveries in the exploratory data analysis. The most important points to recall were that the highest correlation with the response variable observed in the fields: **'avg_rooms'**, **'pct_pre1940'** and **'pct_lowIncome'**. These were the most 'normal' data points with the clearest linear relationship to the response. We can observe their relative influence remains high in this week's assignment as well.

Feature Importance in gradient boosting model 1d



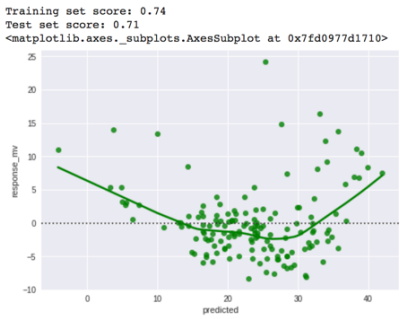
We can observe this fact here in the plot on the left, which is the 'feature importance' mapping for this week's final recommended model (1d). The recommended model applies gradient boosting to determine the best relationship of the variables in a decision tree model. The point to note is that other features like distance to center, crime rate, tax and parent teacher ratio take on more importance relative to other models.

In contrast, the more simplistic linear models applied previously in assignment 3 were not able to get to the training/test scores that we see using the gradient boosting technique. This may be due to the fact that the linear models could not weigh these additional variables without making the model overly sensitive to changes, resulting in over-fit.

In this assignment, I start with the assumption that we will not use the neighborhood classifier and I build four model examples from two sets of data. The first set of data is simply the original data without any transformation.

The second set of data is a log transformation. For each data set, I investigate four modeling techniques: a regularized Ridge model, a decision tree model, a random forest model, and finally a model that uses gradient boosting. The residual plot to the right is an example of where we start our model exploration – a simple Ridge model with light constraints. The general performance of the model is poor with a training/test score in the range of .75. As we will see, this falls short of what we are able to achieve using more complex methods, and is only a point of reference to improve upon.

Regularized techniques are simply a 'starting' point



Training/Test scores ~71 and curved residual plot and high range of values. Analysis - poor fit

[1] David A. Belsley, Edwin Kuh, and Roy E.Welsch. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Wiley, New York, 1980.

[2] Géron, A. 2017. *Hands-On Machine Learning with Scikit-Learn & TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* Sebastopol, Calif.: O'Reilly. Source code available at https://github.com/ageron/handson-ml

Here is a table showing the results of the modeling methods and evaluation of fit. In the end, the highlighted model (1d) results in the best balance between training and test scores, is the most simple and has a good diversification of influence across multiple variables, as shown above in the 'feature importance' plot.

Data Format	Model a: Ridge	Model b: Decision Tree	Model c: Random Forest	Model d: Gradient Boosting
Model 1: Original data (no Δ)	12 features used Training = .74; Test = .71 **Poor fit/overly complex*	Max Depth = 5 Training = .93; Test = .84 **improved fit – 'avg_rooms'*	Max Depth = 4 Training = .92; Test = .87 **good fit – 'pct_lowIncome'*	**Max Depth = 2** **Training = .95; Test = .88** ****'best' fit – balanced features***
Model 2: Log transform	12 features used Training = .78; Test = .81 **Poor fit/overly complex*	Max Depth = 4 Training = .87; Test = .66 **over-fit – not good*	Max Depth = 4 Training = .91; Test = .85 **better fit – 'pct_lowIncome'*	Max Depth = 4 Training = .94; Test = .86 **'best' fit – balanced features*

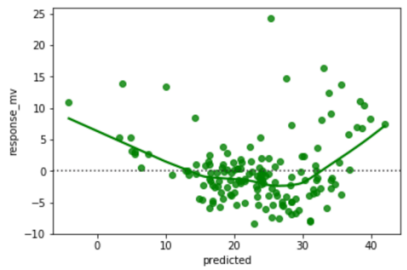
Programming overview:

Both model 1 and 2 use the SciKit Learn 'train_test_split' with a 70%/30% split. I then develop each model variation (a to d) using that same split of data for the model fit and testing in each model. The only difference is in the preparation of Model 2, where I apply a log transformation. I then print the training & test 'score' and plot the residuals using seaborn residplot.

```
>> print("Training set score: {:.2f}".format(model.score(X_train,y_train)))
>> print("Test set score: {:.2f}".format(model.score(X_test,y_test)))
>> sns.residplot(y_test['predicted'], y_test['response_mv'], lowess=True, color="g")
```

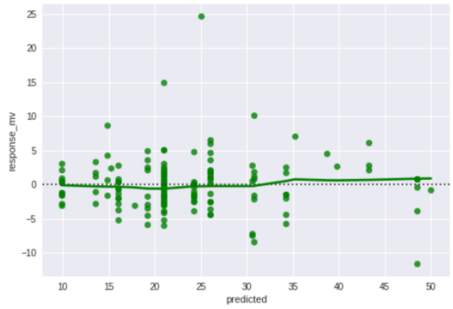
a) Ridge(alpha=.001, solver="cholesky")

```
Training set score: 0.74
Test set score: 0.71
Number of features used: 12
<matplotlib.axes._subplots.AxesSubplot at 0x1c17d64eb8>
```



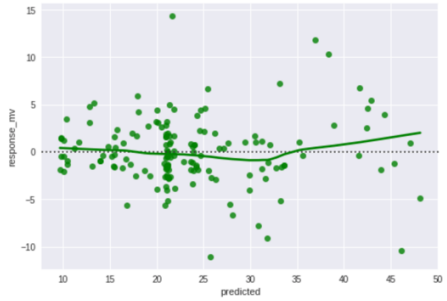
b) DecisionTreeRegressor(random_state = 9999, max_depth = 5)

```
Training set score: 0.93
Test set score: 0.84
<matplotlib.axes._subplots.AxesSubplot at 0x7f5b74da9048>
```



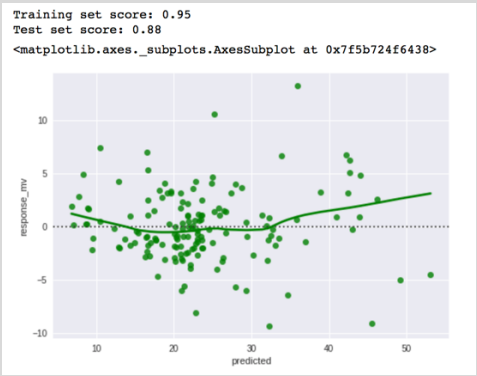
c) RandomForestRegressor(max_depth=4, random_state=0, n_estimators=100)

```
Training set score: 0.92
Test set score: 0.87
<matplotlib.axes._subplots.AxesSubplot at 0x7f5b74dbe7f0>
```



d) GradientBoostingRegressor(max_depth=2, n_estimators=120, random_state=42)

```
Training set score: 0.95
Test set score: 0.88
<matplotlib.axes._subplots.AxesSubplot at 0x7f5b724f6438>
```



(recommended 'best fit' model)

Note: A shared version of the python notebook used to create this analysis can be found at Google CoLab: https://colab.research.google.com/drive/1pN81K22-HYHZPMiwI28ZmMnWqRFwFlew

Conclusion: In removing the classifier 'neighborhood', the initial regression models performed poorly. By using decision tree, random forest and gradient boosting, we were able to bootstrap the data and develop a model that performed well on both training and test data (.88 test score). Therefore, model (1d) using gradient boosting without transformed data is the recommended model. It has the highest performance scores, does not over-rely on any one independent variable, and residual plotting is reasonably flat in comparison to the other models.