Project 2 Evaluation Rubric

Instructions:

1. Rename this file with your city and name last name first i.e. Project 1 Evaluation Tool - LA - Cook\_Joshua.docx
2. For each of the selected questions, compare your code with the given solution.
3. It's OK if the path you took to get there was different, but the result should be the same.
4. Evaluate yourself based on:
   1. your code
   2. conceptual understanding
   3. what you could've improved
5. **As this project is MUCH more open-ended, your results may differ significantly from ours. Evaluate yourself on the intent of the question and response, and your ability to execute.**

Keep in mind:

* Self-evaluation is important, and learning how to identify issues yourself is critical.
* The goal here is for both you AND an instructor to evaluate your work. You should be thinking critically about your answers and compare your self-evaluation with instructor feedback.
* Your comments on your work are in an integral part of how we will be providing feedback. The more detailed that you are with your discussion of your work, the more feedback we will be able to give you to help you grow.
* **Our answers should not be thought of as gospel**. According to the Zen of Python, “There should be one-- and preferably only one --obvious way to do it. Although that way may not be obvious at first unless you're Dutch.” Keep in mind that none of your instructors are Dutch and while we do our best to adhere to [PEP 8](https://www.python.org/dev/peps/pep-0008/), we may do things that are less than Pythonic.
* You should use the [Skitch](https://evernote.com/products/skitch) tool or some other similar tool to take screenshots of your work that you paste into this document as we have done with our answers.

Submit this doc on Sunday by 11:59 PM PST.

Project 2 Evluation Rubric

**3b. Fill missing data.**

**4a. Benchmarking**

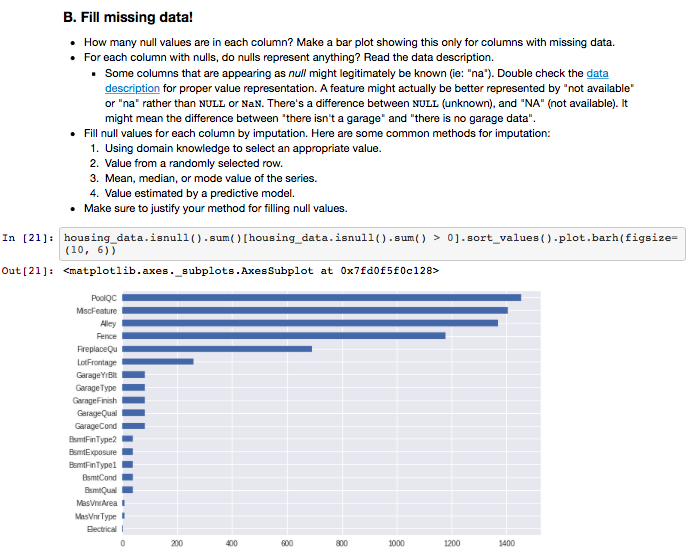
**4c. Cross-validated models**

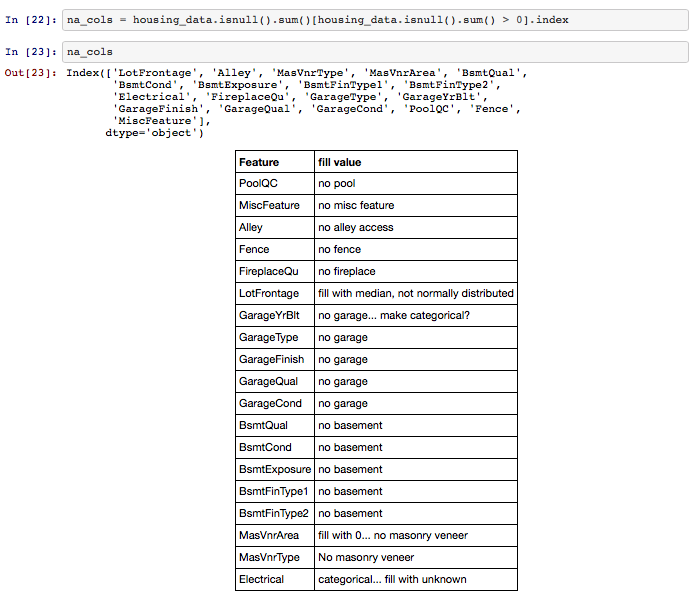
**4D. Model Selection**

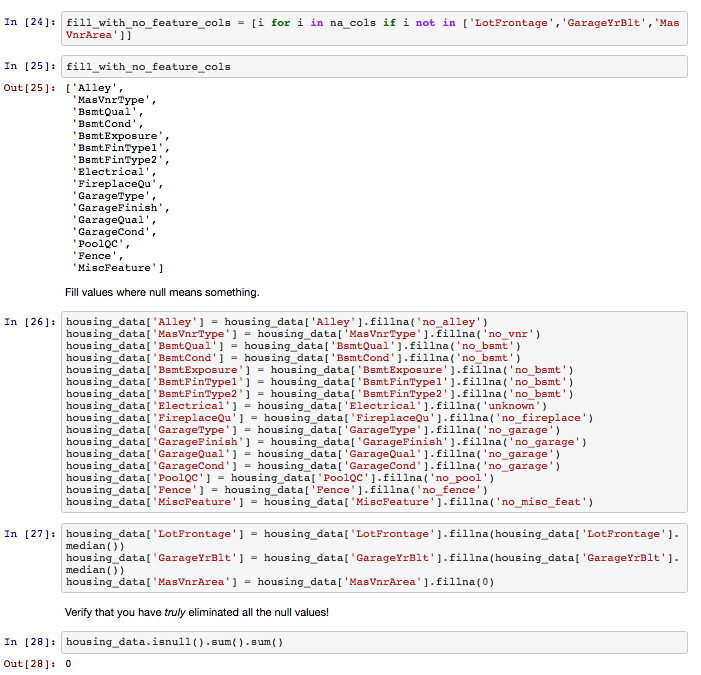
**High-Level Evaluation**

**3b. Fill missing data**

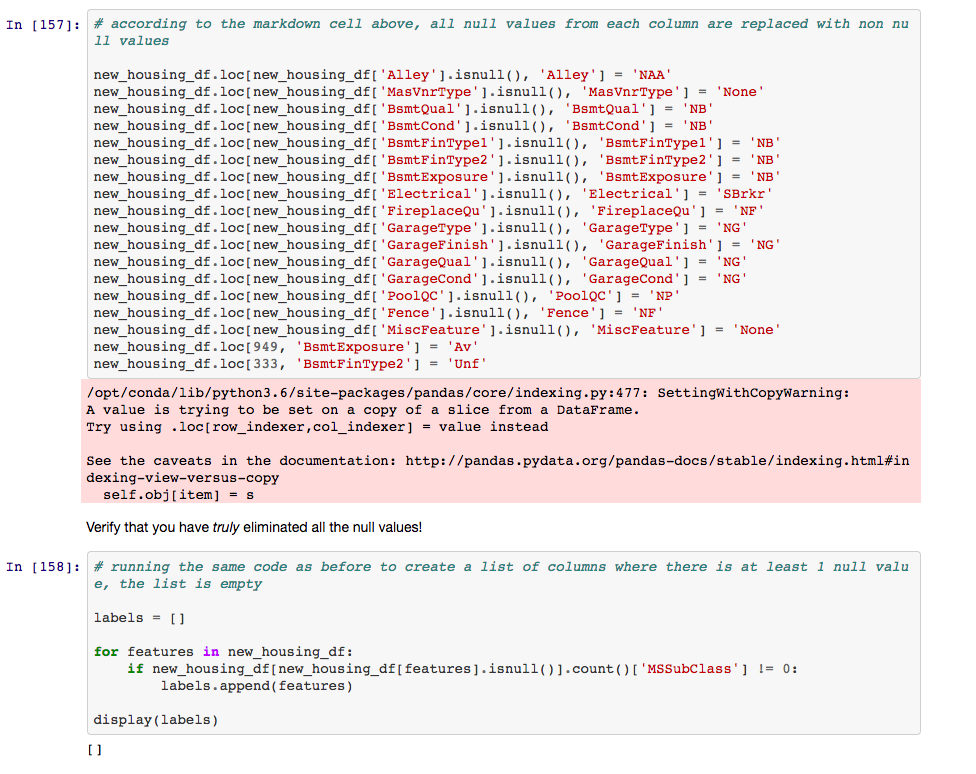
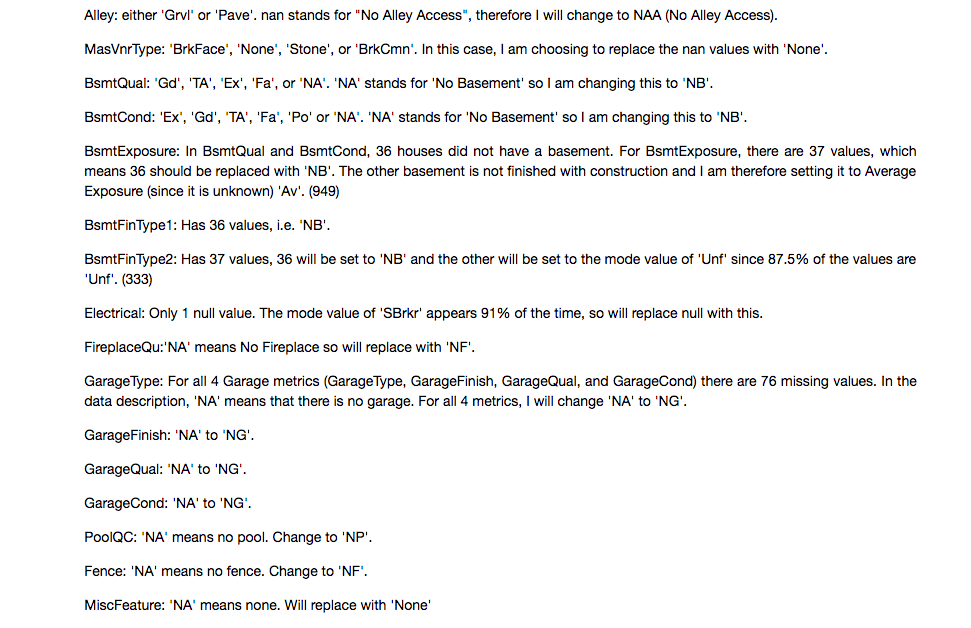
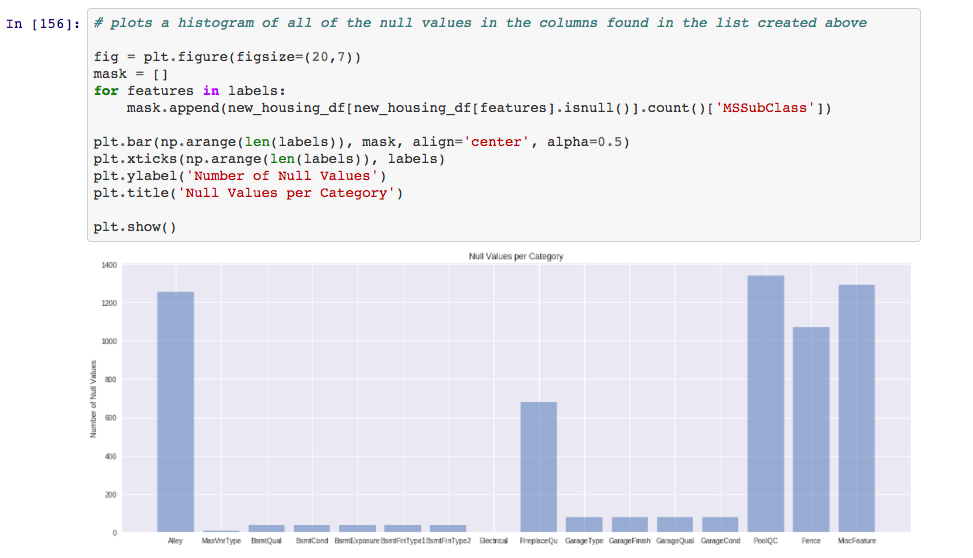
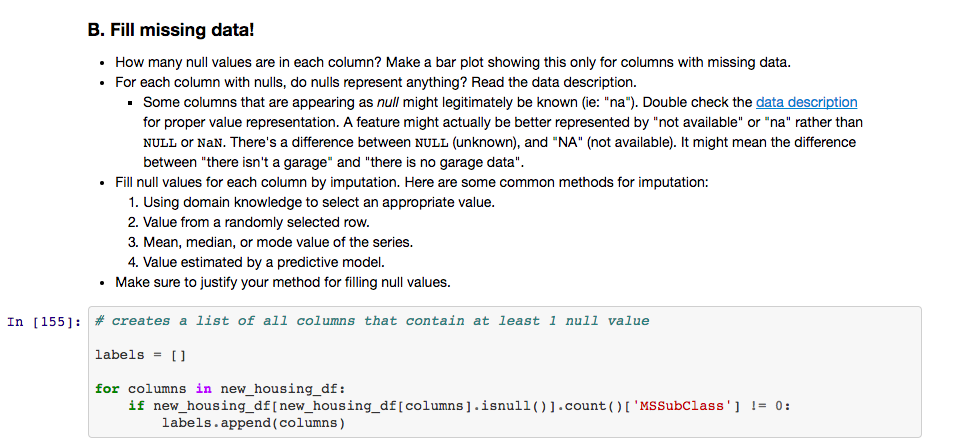
***Our Answer***

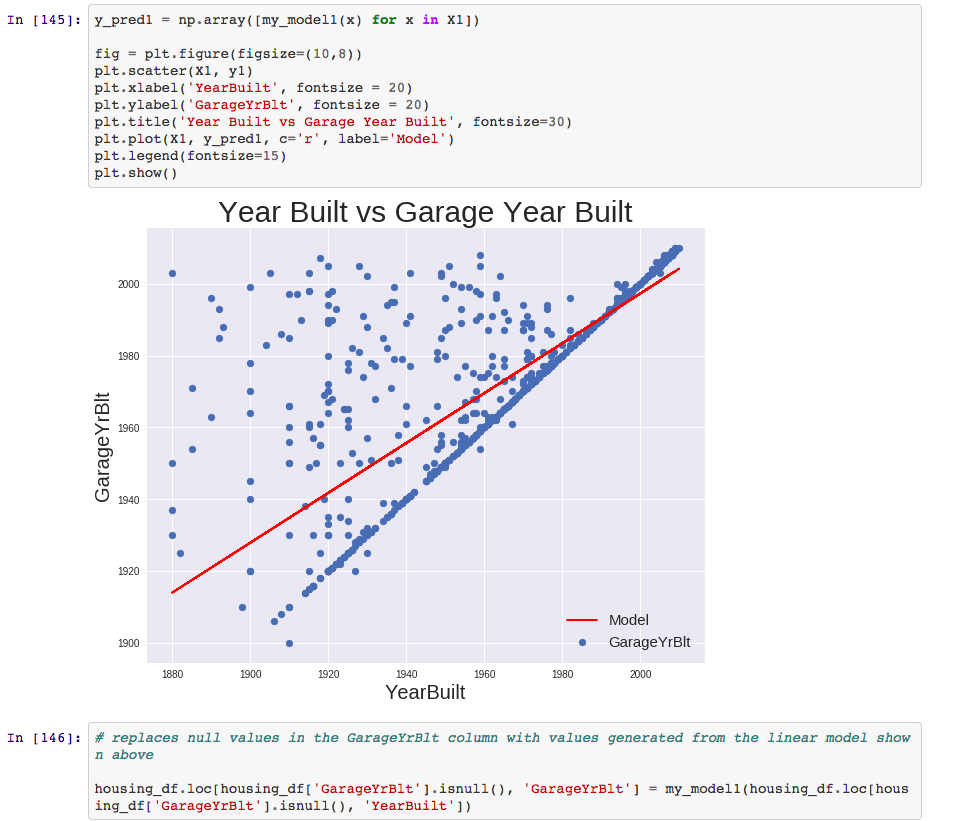
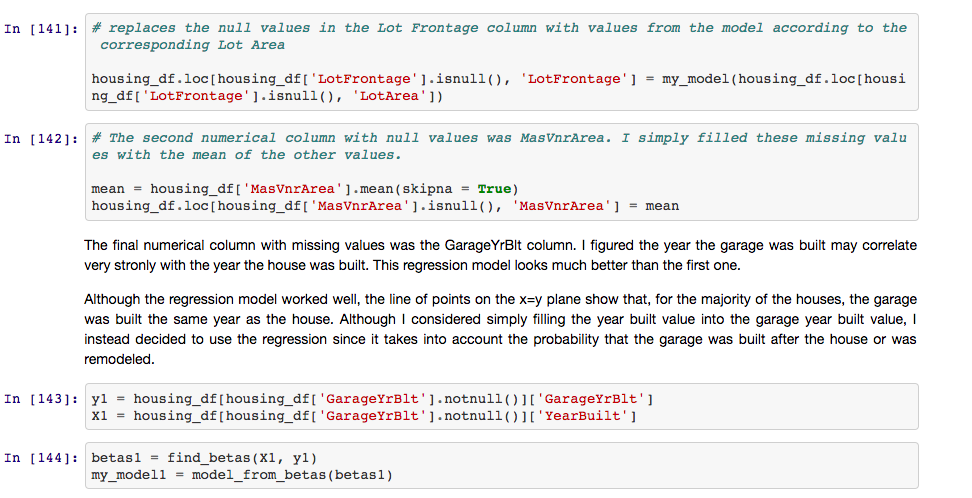
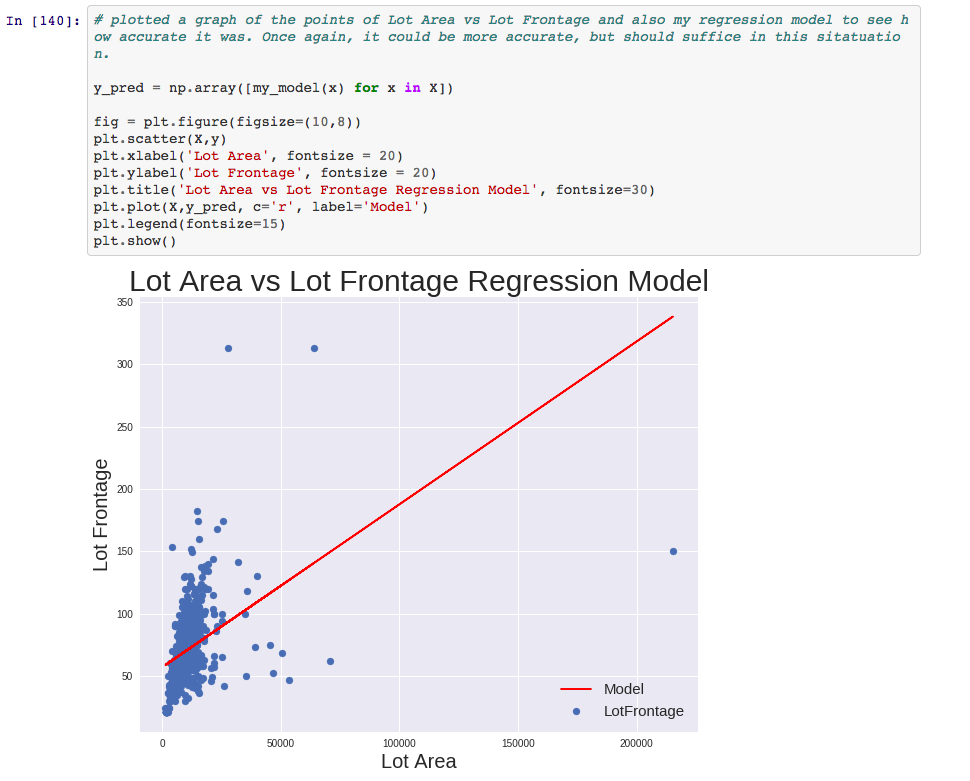
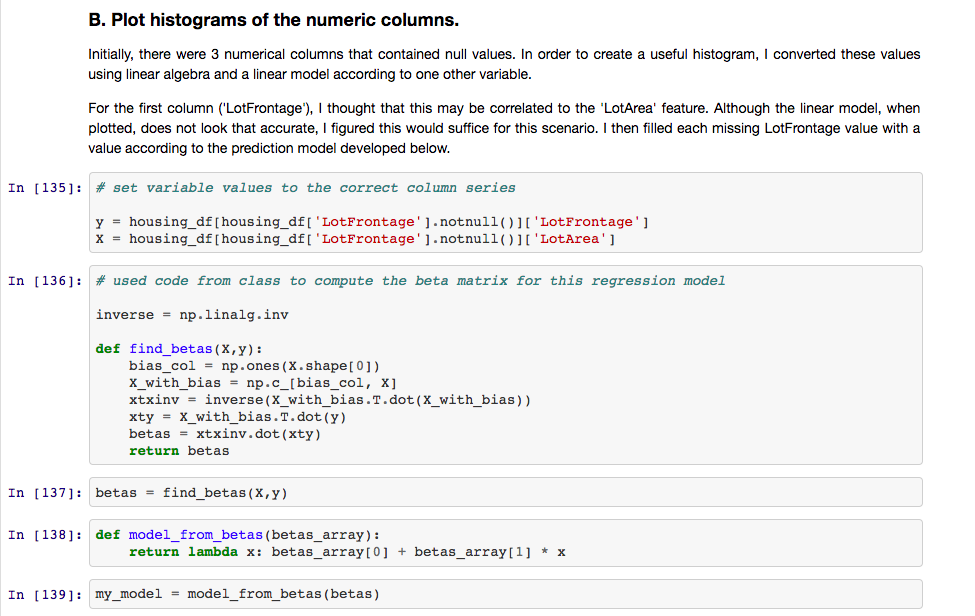






***Your*** ***Response***

**

***Additional Code to Explain My Solution***

***Your Rating for your response: (2)***

( 3 - I killed it, 2 - I did pretty good, 1 - I struggled)

***Why you selected this rating***

Although my code accomplishes the task, the solution outputs are more organized (bar plot is sorted and created a table to show the fill values for each column), and I replaced the null values of the columns using .loc instead of .fillna(), where .fillna() is a more efficient way to accomplish the task.

My solution uses a for loop to loop through all columns of the dataframe. If the number of isnull() values is not zero (if there are null values), this column name is appended to the labels list initialized outside of the for loop. Another for loop is then used to created a list of the count of null values in each column. Finally, a bar plot is created using the list with the column names (labels) and the list with the number of null values in each corresponding column (mask). My thought process with regards to how to replace the null values in each column is explained, and then the null values are replaced as discussed. I use the same for loop that I initially used to create the list of columns that contain at least one null values, and I create another list of columns that contain at least one null values. This list is rightly empty.

My solution does not include the columns GarageYrBlt nor LotFrontage. This is because I filled the null values in these numerical columns before I plotted histograms of each numerical column in question 2B (Additional Code to Explain My Solution). Using beta matrices and linear models taught in class, I created linear models of each feature against a feature I thought may have a connection to the desired feature. As you can see, the Lot Area vs Lot Frontage linear model is not much better than having just used the median value, as the solution code does. However, the linear model created for GarageYrBlt vs YearBuilt looks like a relatively accurate model, and replacing the null values with this model could be a better option than just using the median value.

The solution code does a better job of organizing a visual representation of which columns have how many null values as well as how each column’s null values are replaced. I think my solution could be improved by sorting the bar plot of the null value counts from high to low or vice versa to create some organization in the plot. The table the solution code uses to depict the fill values for each column also helps to make the code more readable. For the actual action of replacing the null values, my code and the solution code use different ways to replace the values, mine using the .loc function for dataframes, and the solution code using the .fillna() function. The .fillna() function, since it is specifically designed for this type of use, is a more efficient way to replace the values than what I used.

***Our Rating for your response: (3/2/1)***

***2***

***Why we selected this rating***

***The Good:***

***Your writeup in markdown of your column-by-column reasoning. Very clear very nice.***

***Imputation of continuous nulls***

***The Bad:***

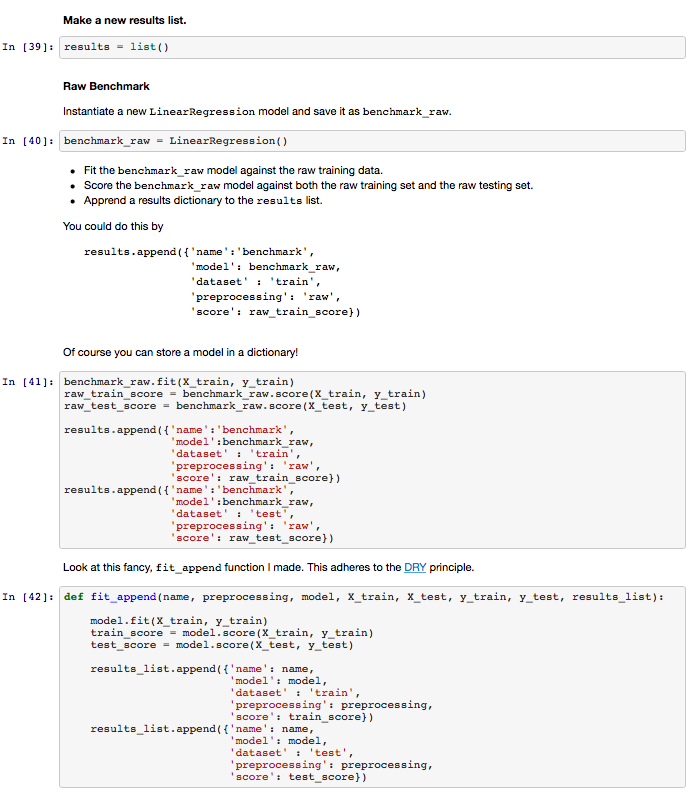
***Using loops instead of pandas features to check for nulls***

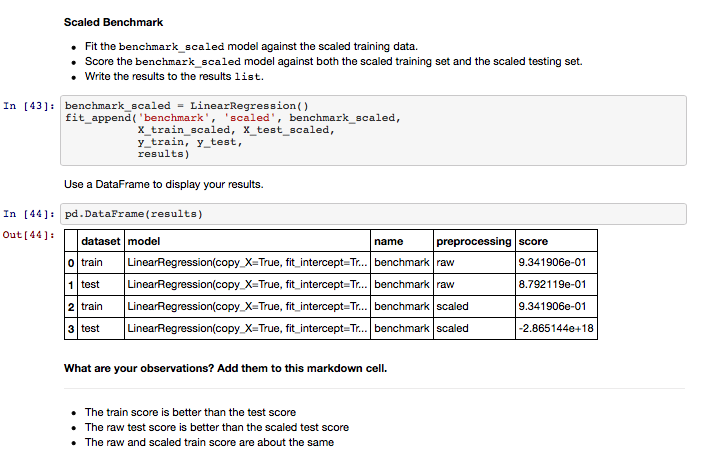
***Imputing LotFrontage with LotArea seems questionable. What was your R^2. That’s the kind of number that should justify that choice.***

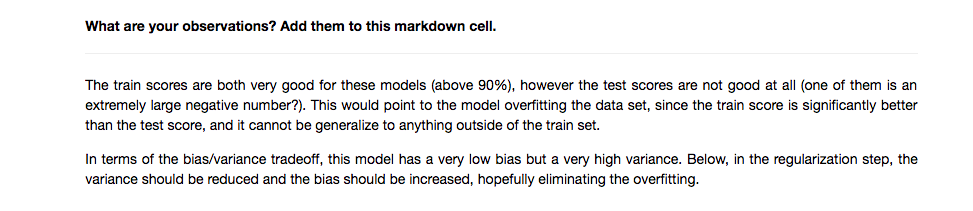
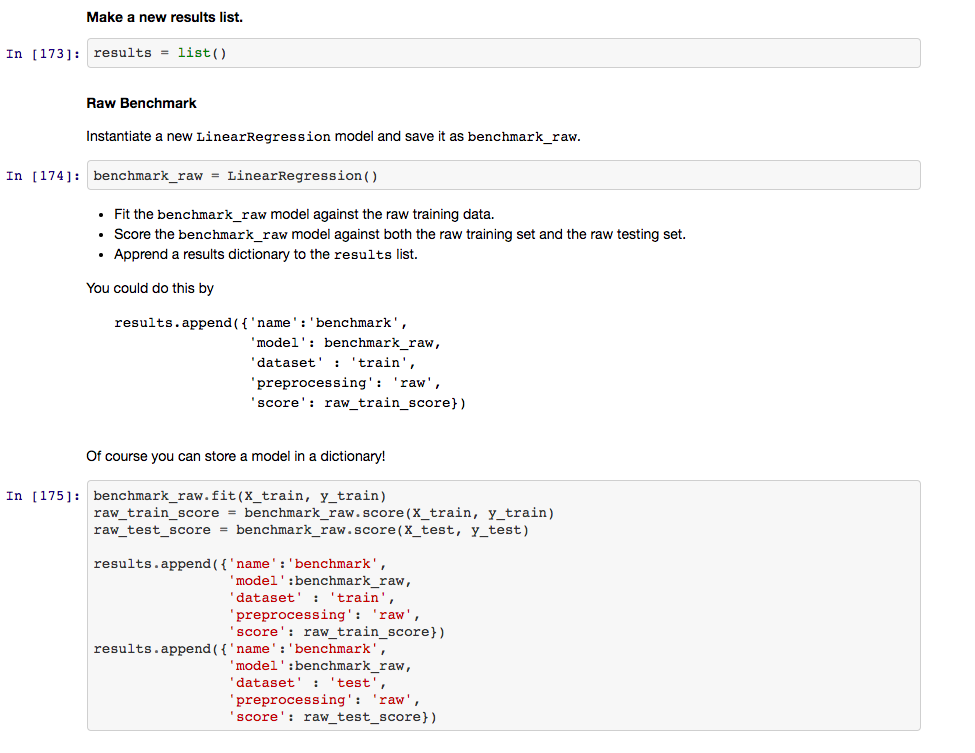
***Overall good work on this question. You’re putting in the grit which is what gets it done in the end.***

**4a. Benchmarking**

***Our Answer***





***Your*** ***Response***

***Your Rating for your response: (2)***

( 3 - I killed it, 2 - I did pretty good, 1 - I struggled)

***Why you selected this rating***

My solution uses virtually the same code as the solution code, however the solution code is created in a function, and the function is then called throughout the project. Although for this specific question, the function does not create more efficiency, for the rest of the project, the solution code has the option to call this function for all of the following scores that need to be calculated, which works more efficiently than my code.

The test score for the benchmark raw is also very different between the two solutions. My solution does not have a high test score for either raw or scaled, with the benchmark raw test score being only 0.514. In contrast, the solution code’s benchmark raw test score is 0.879. This could be for many different reasons (how we filled null values, outlier detection, etc.), however these low test scores coupled with high train scores make me think my model is overfitting the dataset, while the solution code’s model looks like it is fitting the dataset well.

The large negative test score means that the benchmark model fit with scaled data does a very poor job of fitting the dataset. Since the score is calculated using the R^2 metric, a 1.0 score is the highest possible score, and a 0.0 score is equivalent to having just predicted the mean value for all y values (Sale Price in this scenario). The benchmark scaled test score is a large negative number, which can be interpreted as if I had predicted the mean Sale Price for every home, that would be a significantly better model than the model created.

***Our Rating for your response: (3/2/1)***

***3***

***Why we selected this rating***

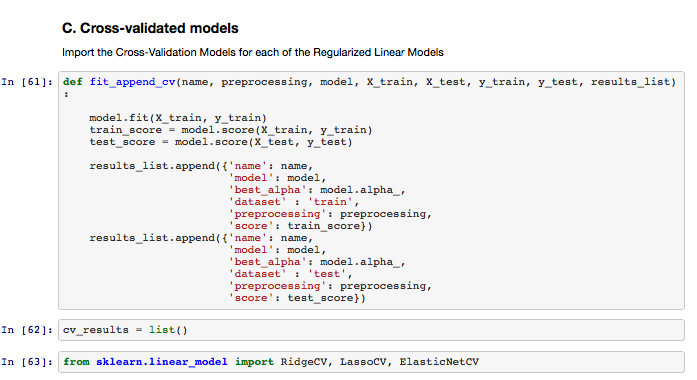
***Clean code. Thoughtful interpretation. What else could I want?***

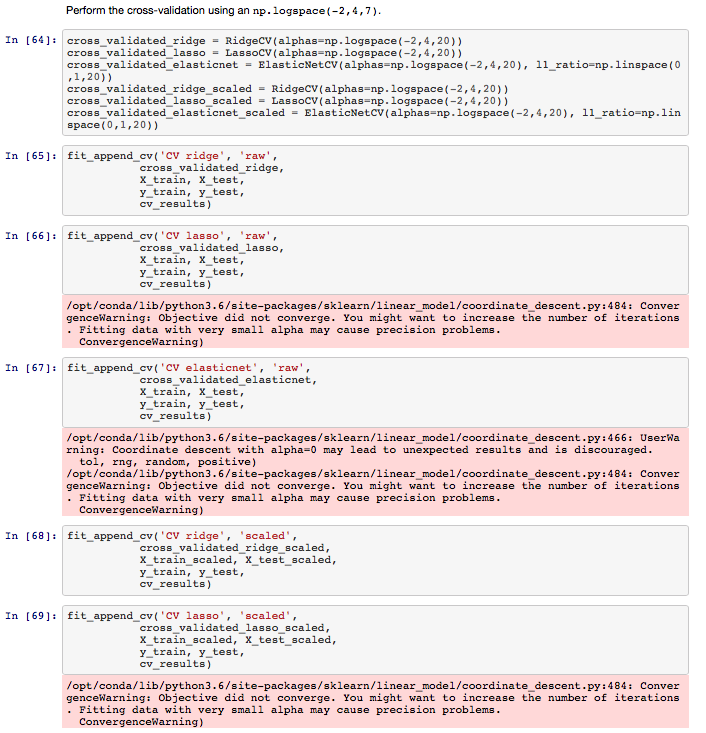
***With regard to the horrible R^2 on the scaled tests set, I’m really not sure what’s causing that. I want to investigate more myself. It could be that scaling the data blows up outliers in low-variance columns. Interesting very interesting.***

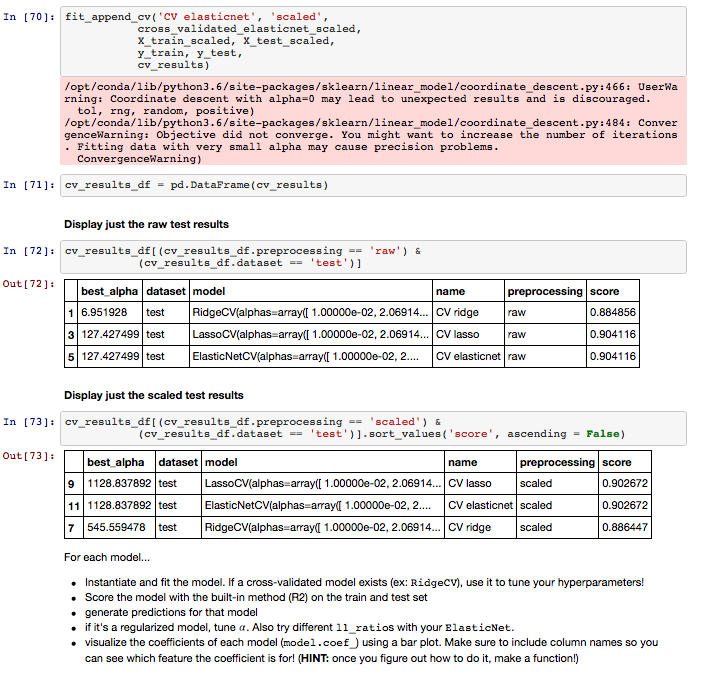
## 

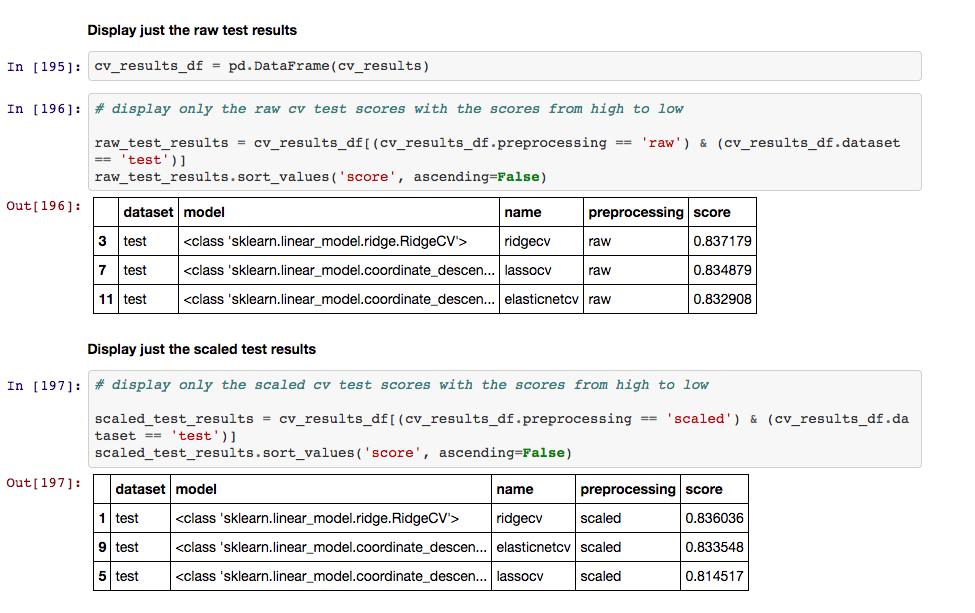
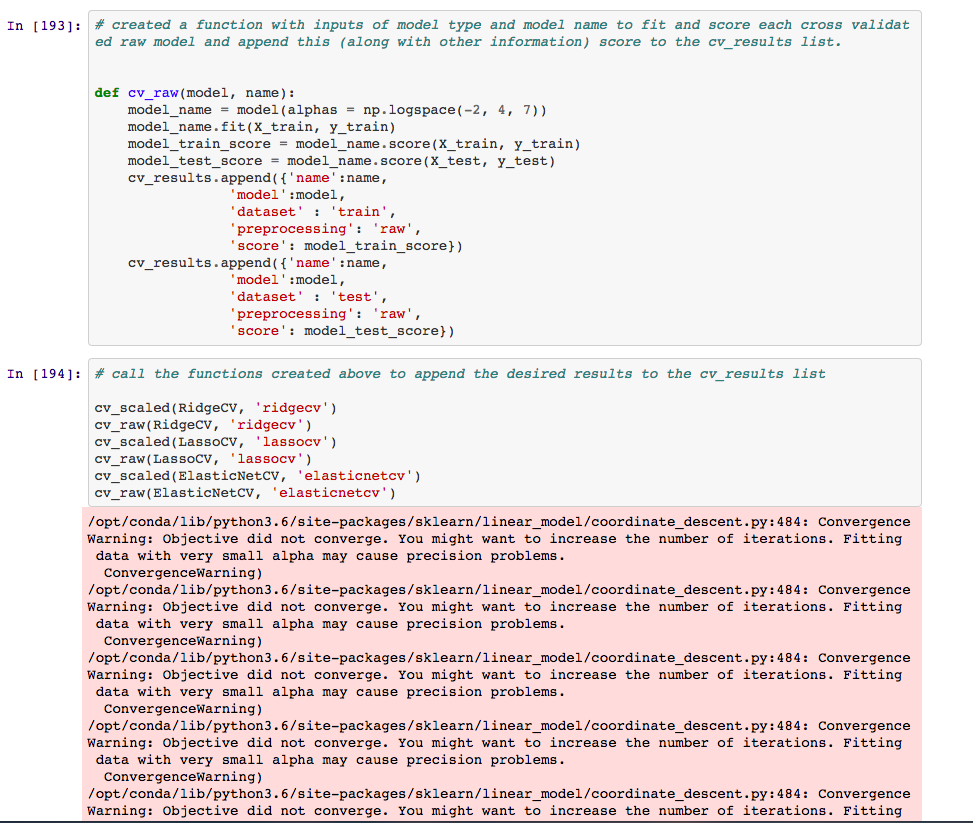
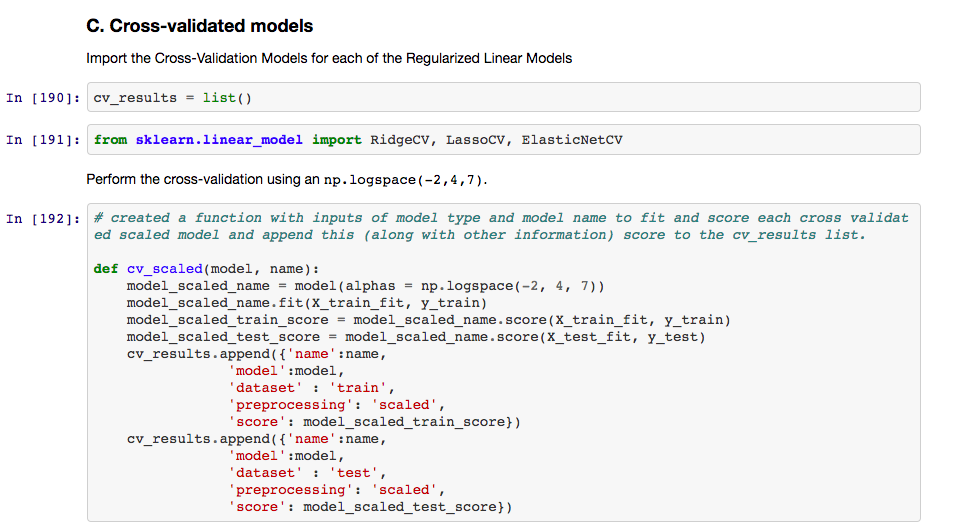
**4c. Cross-validated models**

***Our Answer***







***Your*** ***Response***

***Your Rating for your response: (2)***

( 3 - I killed it, 2 - I did pretty good, 1 - I struggled)

***Why you selected this rating***

My code is written efficiently and is readable. Although the solution code uses the function created in earlier steps (which is overall more efficient than how I approached the project), for this step I have created functions as well and use these functions to create the necessary lists.

Initially, I attempted to create one function for all models to fit and append the scores to a list, however I was having trouble with differentiating the cross validation models vs the initial models, and how to fit the model using the np.logspace( -2, 4, 7 ) for cross validation, but not do that for the non-cross validated models. My function was initially defined as follows:

def model\_raw(model, name, cv=True):

if cv:

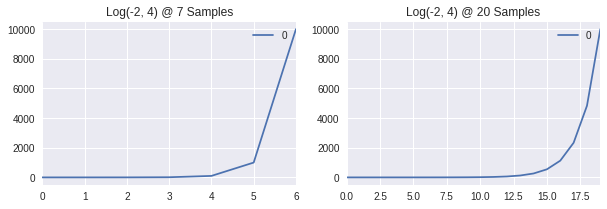
model\_name = model( alphas = np.logspace(-2, 4, 7))

else:

model\_name = model()

When this code was run with cv=False input (for the non-cross validated models), the function worked properly. However, when the code was run without a cv input or with cv=True as the input, the cross validated models and their scores would be appended to the list two times, one score with the np.logspace(-2, 4, 7) used in the model, and one score without this np.logspace function. Somehow, the function was going through both if statements, but I could not properly debug the issue so instead I created two different sets of function calls, one for cross validated and one for not cross validated models.

Another efficiency problem I had was that I created separate functions for raw data vs scaled data models. This, again, was done more efficiently in the solution code by creating one function with many input parameters.

Overall, my test and train scores are acceptable and within a range of what I would expect to see, however not quite as good as the solution code. This could be for many different reasons, the most likely being the different ways we eliminated outliers and/or the different ways we filled missing values in the dataset.  


***Our Rating for your response: (3/2/1)***

***2***

***Why we selected this rating***

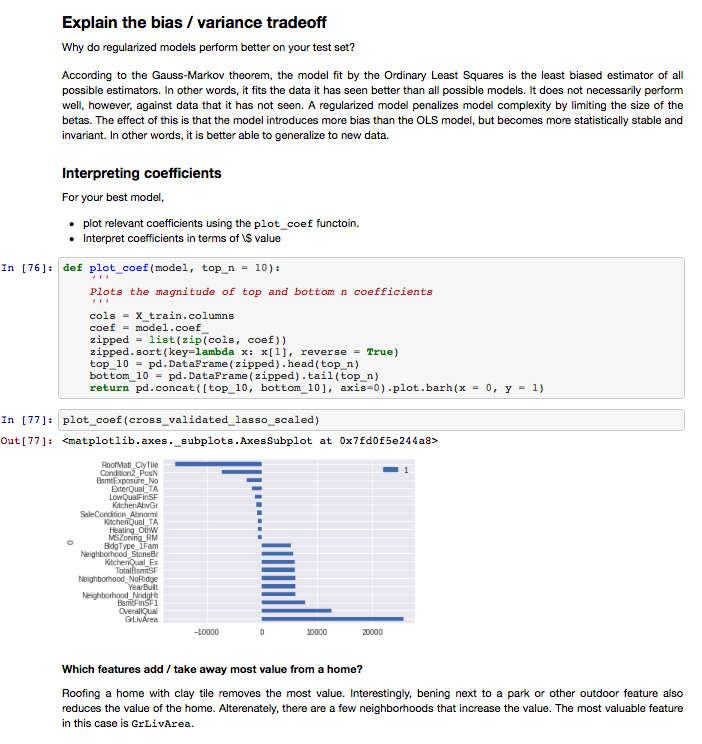
***I think you’re self-assessments are totally on-point.***

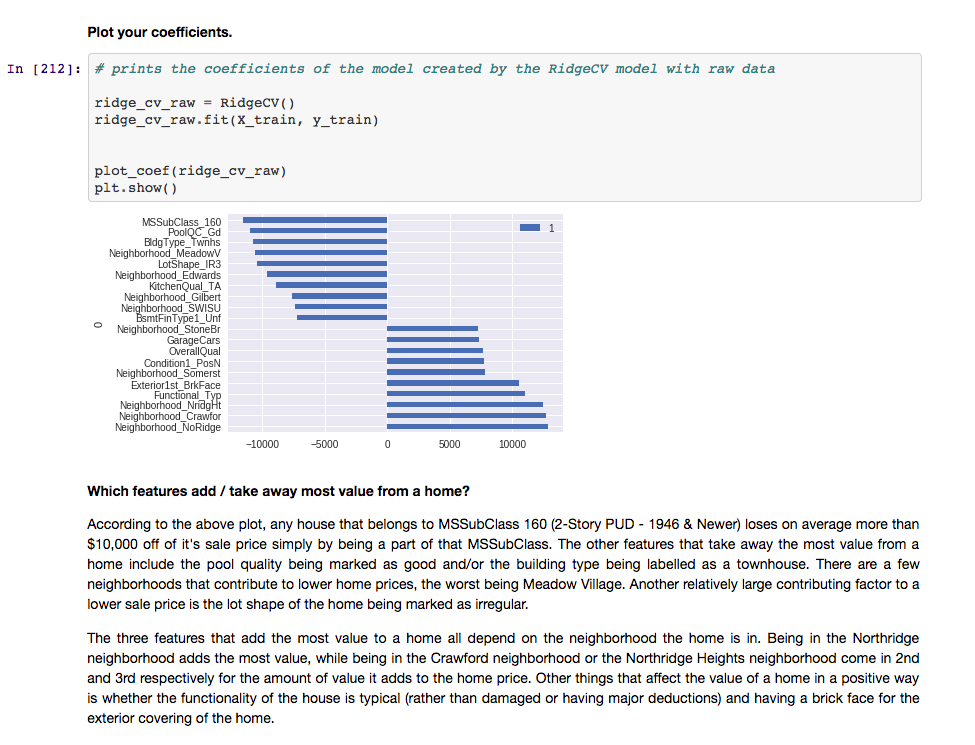
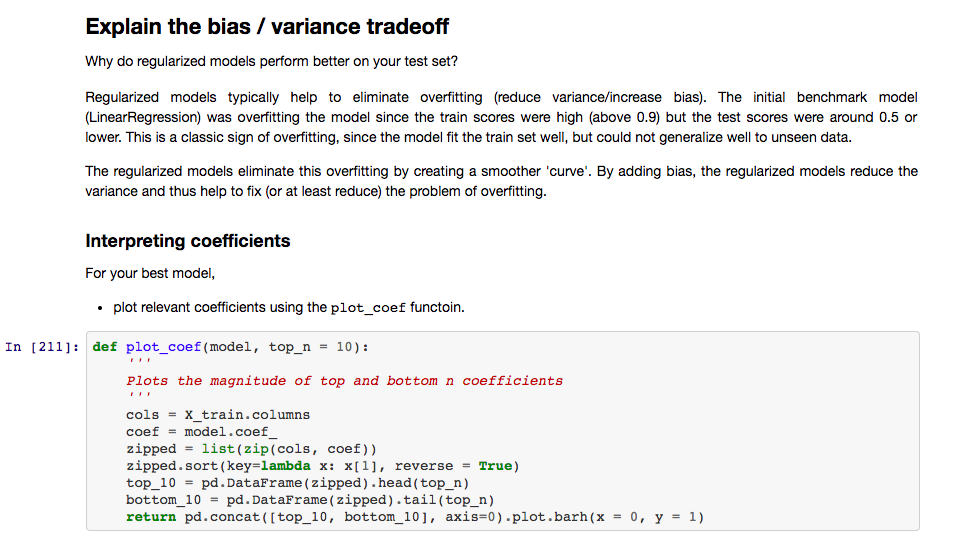
***My main gripe is with the separate functions for raw and scaled data. That is a textbook case where that functionality could be abstracted away into a single function.***

## 

**Model Selection: Bias/Variance Tradeoff and Plotting Coefficients**

***Our Answer***



***Your*** ***Response***

***Your Rating for your response: (3)***

( 3 - I killed it, 2 - I did pretty good, 1 - I struggled)

***Why you selected this rating***

According to my model and my error measurements, these were in fact the coefficients for each feature, and the discussion of what added or subtracted value from a home price is well thought out. The cross validated ridge model fit with raw data was my highest performing model, whereas for the solution code, the best model was the cross validated lasso model fit with scaled data. The lasso model uses less weight in a majority of the features, with a very high weight placed on a few of the features, both in a positive and negative direction for sale price. The model used in my solution, the ridge model, has a more evenly distributed weight among the coefficients.

My model places a much higher weight on the neighborhood the house is located in and the correlation between that neighborhood and the price of the home.

Although I think the solution code has a better model than what I have, I gave myself a 3 on this question, because I used many different error metrics (including RMSLE the error metric used on Kaggle to decide the winner of this problem) to determine the best model, and have a good discussion about the different coefficients and which features have large positive and negative weights for Sale Price.

***Our Rating for your response: (3/2/1)***

***3***

***Why we selected this rating***

***Props for using alternative error metrics and your interpretation of the coefficients***

***Lets talk nitpick*** “Regularized models eliminate overfitting by producing a smoother curve”***.***

***I think you were referring to the affect of regularization on polynomial regression — the coefficients of higher-order, unnecessary terms are reduced producing a smoother curve. That is just one case of regularization. In multi-linear regression, predictions are a hyper-plane, a surface that is always smooth. What regularization does is diminish the model’s sensitivity to certain features by penalizing the magnitude of parameters.***

**High-Level Evaluation**

***One thing I did very well on this project was …***

Last project, I was docked points for not assigning readable variable names, and also for not generating programmatic output. I specifically focused on these things in this project, and I think I did a better job of creating functions whenever possible rather than repeating code. I also focused on naming variables in a more descriptive way, however I could still improve.

***O******ne thing that I’m not clear on and would like to know more about is …***

What to look for when making decisions with regards to filling null values, eliminating outliers, as well as choosing important features. I have a quantitative background so I have a good understanding of numbers and data. However, it would be very helpful for me to hear a professor’s perspective and rationale with regards to choosing between filling missing values using a linear regression model, the median, the mean, or another type of model as well as where to “cutoff” the outliers of a dataset.

***If I was starting this project over from the beginning knowing what I know now, one thing I would do differently is ...***

I would try to create one “master” function to fit and append models and their scores to a results list (the way the solution code does). I did try to make functions, and I even attempted to create a more general function than what I ended with, however using more parameters and creating a single function makes the code much more readable and efficient, and also eliminates repetitive coding for me.