Problem Statement

Our project focused on determining whether we can predict the value of owner-occupied homes in the city of Boston and understand features which are important to homelessness in the Boston area.

Exploratory Data Analysis (EDA)

Aim I. Analyze trends in homelessness in America from 2007-2016.

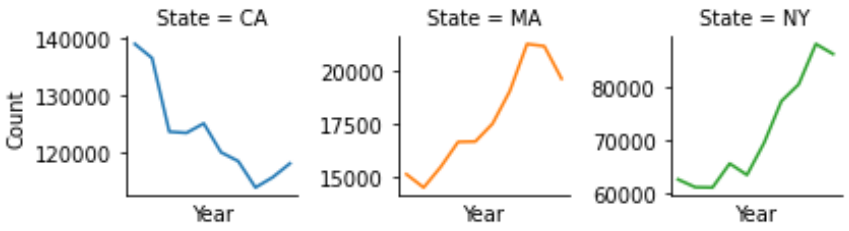
Exploratory data analysis for the "2007-2016-Homelessness-USA.csv" file began by loading it into a Pandas dataframe and taking notice of the variables, their data types, and whether the set is missing any values. Here is a screencap of the first 5 rows:

	Year	State	CoC Number	CoC Name	Measures	Count
0	1/1/2007	AK	AK-500	Anchorage CoC	Chronically Homeless Individuals	224
1	1/1/2007	AK	AK-500	Anchorage CoC	Homeless Individuals	696
2	1/1/2007	AK	AK-500	Anchorage CoC	Homeless People in Families	278
3	1/1/2007	AK	AK-500	Anchorage CoC	Sheltered Chronically Homeless Individuals	187
4	1/1/2007	AK	AK-500	Anchorage CoC	Sheltered Homeless	842

The dataset contains 6 variables, two of which are numeric and four that are strings. A little background research reveals "CoC" to mean "Continuum of Care"-- this is essentially a regional partition that the US Housing Administration uses to divide resources for the homeless. Each region makes estimates for how much of each type of homeless are within their sphere of responsibility. The next step in the EDA process was to change the Year and Count columns into appropriate Pandas data types. Year is converted to a datetime object and Count is converted to an integer.

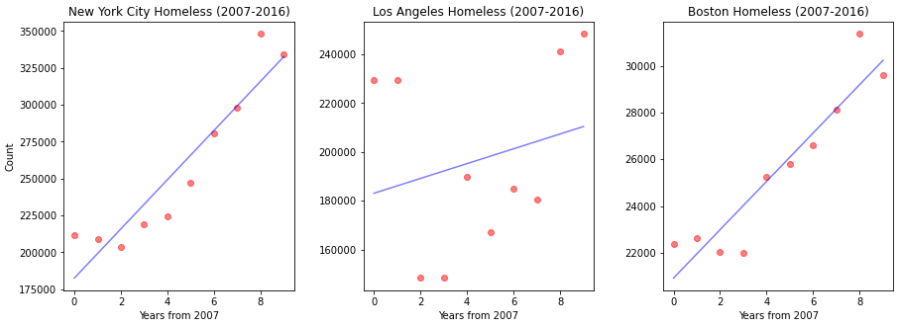
At this point the info(), describe(), and nunique() methods are used to probe deeper into the dataset characteristics. The set is found to be complete with no missing values. There are 10 years captured in the data, from 2007 to 2016. There are 54 "States", which include the District of Columbia, Puerto Rico, Guam, and the Virgin Islands. There are also 414 CoCs, and 42 "Measures". The Measures are a breakout of the types of homeless people within each CoC. Counts for each CoC are found to vary wildly; The minimum is 0, maximum is 75323, the mean is 332.28, and the median is 52.

The next step in the EDA was to try and understand general trends of homelessness for each state. There is a "Total Homeless" category under Measures that gives a sum of all homeless people within a CoC. Displayed below are the plots for combined homeless people in Los Angeles, California, Boston, Massachusetts, and NYC, New York (the rest of the states can be seen in the notebook).



General trends here show some interesting features between the three states. California saw a large decline in homelessness throughout much of the decade, except for the last couple of years.

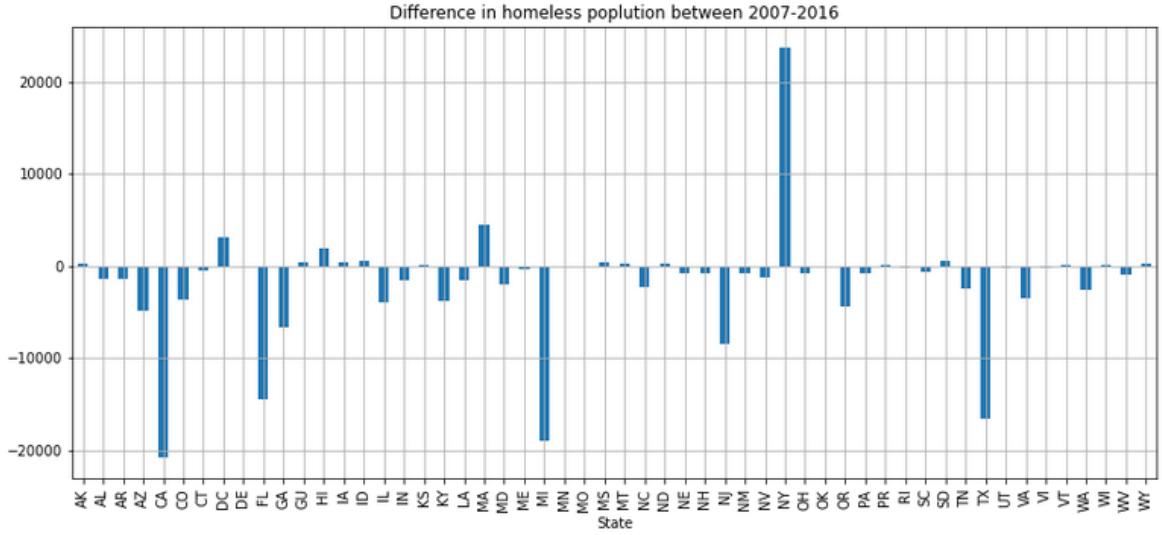
Massachusetts and New York saw the opposite trend, which is interesting considering these states contain some of the largest coastal cities (assumption being that large cities typically see large homeless populations). Perhaps there are regional factors in MA and NY that were or were not present in CA during the same period.



We followed up on this analysis by comparing the homelessness of Boston to other major cities. Here, we looked at the homelessness data from New York City, Los Angeles, and Boston and compared the three. We fitted the data on a Linear Regression model to obtain the trends in the homeless population from 2007-2016 in these three cities. From these three models, we observed that the total homeless population had increased in each city over the 9-year period.

The R-squared for New York City, Los Angeles, and Boston were 0.87, 0.059, 0.87, respectively. These results show that NYC and Boston show linear trends of homelessness increasing. However, Los Angeles homelessness count does not follow a linear model. The slope for New York homeless was 16687 and the slope for Boston homelessness was 1035. Thus, the rate of homelessness was faster in New York than in Boston.

Another chart that was created for this EDA was the difference in homelessness throughout the decade for each state. Unfortunately, the dataset does not contain state (or CoC) general populations during the timeframe. So this chart couldn't be normalized so that each state can be compared more directly.



Over the time period, it is clear homelessness increased considerably in New York, and decreased considerably in California, Florida, Michigan, and Texas. What could be the causes of such large disparities? Policy failures/successes? Economic factors? Unfortunately, we realized that this dataset does not include enough information to develop a prediction model to answer any of these questions at the state or local levels. So it was decided to disregard the national homelessness data, and focus on the Boston housing data to complete the project.

Aim II. Predict median value of ownership occupied homes and determine feature importance.

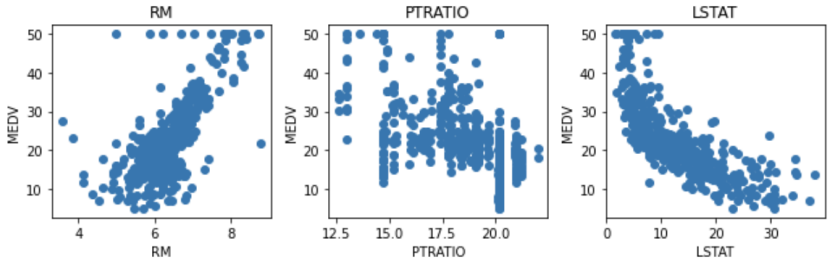
The Boston housing dataset is from data ('boston.csv') collected in 1978 by the U.S. Census Service containing aggregated housing data from various suburbs in Boston, Massachusetts. For our exploratory data analysis, we first loaded the data as a Pandas dataframe and examined the first five rows, as shown below:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

The dataset contains 14 relevant variables. Our dependent variable is 'MEDV', which is the median value of owner-occupied homes in 1000's of dollars, and all over variables are used as predictor variables. We used the shape(), info(), and describe() functions to gain initial insight into the data. The dataset contained no missing values, so no replacement was needed. All variables were numeric except 'CHAS' and 'RAD', which were converted into categorical variables (with 2 and 9 categories respectively). To explore the distributions of each of the variables, we plotted histograms, as shown in the jupyter notebook.

The histograms give us a sense of the range of values in each of the variables. Some are normally distributed (e.g., 'RM'), while others are skewed (e.g., 'DIS'). Our dependent variable 'MEDV' is approximately normally distributed, indicating we may not need to implement transformations in our main analyses.

We also conducted correlations between each of the variables in the dataset to gain an initial sense of which variables may be related to 'MEDV'. From this analysis, we see that 'RM', 'PTRATIO', and 'LSTAT' are highly (r > .5) correlated with 'MEDV', and may be good predictors. We plot each of these variables



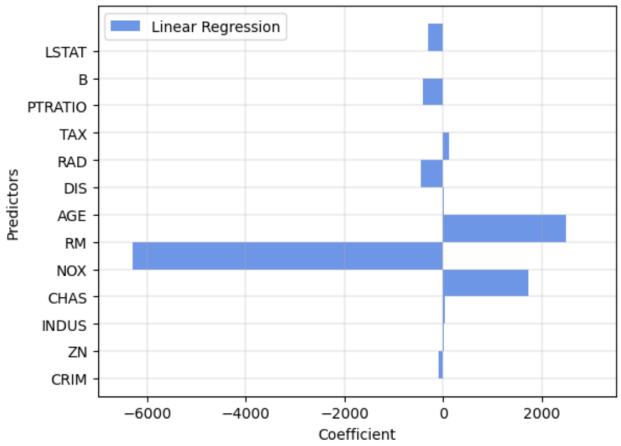
with 'MEDV'. From visual inspection of these plots, 'RM' appears positively correlated with 'MEDV' (i.e., a higher number of average rooms per dwelling relates to an increase in the median value of home prices), while 'PTRATIO' and 'LSTAT' appear negatively correlated with 'MEDV' (i.e., the higher the pupil-teacher ratio by town, the lower the median value of home prices; and the higher the percentage of 'lower status' of the population, the lower the median value of home prices). We will keep these preliminary insights about which variables may be most useful in predicting 'MEDV' in mind for our main analyses and interpretations.

Baseline Model - Linear regression with all predictors

For our initial baseline model, we chose to include all the variables as predictors in a linear regression predicting 'MEDV' (median home value). For this baseline analysis, we first normalized the values of the variables to be on the same scale. We then split the dataset into a training (80%) and test (20%) set. We then fit a linear regression on the training data, predicted on the test data, and computed the mean

squared error (MSE) and r-squared values. The test MSE was calculated to be 26.55 and the r-squared value was calculated to be .73.

We then extracted and plotted the coefficients of the model. The regression coefficients are as follows:
{'CRIM': -90.54945344468192,
 'ZN': 12.435627255449898,
 'INDUS': 35.763042331464725,
 'CHAS': 1731.5468216627505,
 'NOX': -6281.005681631801,
 'RM': 2499.893052429835,
 'AGE': 6.5191960034199,
 'DIS': -445.64695269392973,
 'RAD': 130.5676931901906,
 'TAX': -4.251307121930198,
 'PTRATIO': -403.47982094073575,
 'B': 5.80569827062138,
 'LSTAT': -297.6445318252286}.
For ease of interpretation, we plot these coefficients to the right:

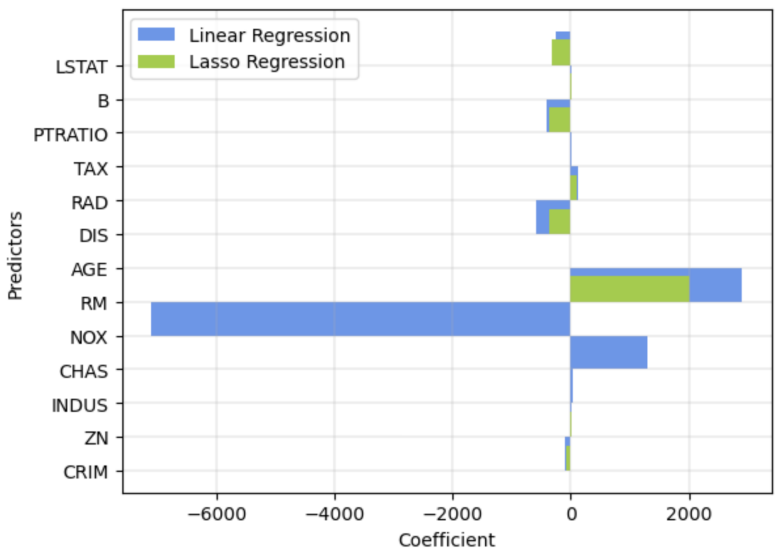


From this plot, we see that some variables (i.e., 'CHAS', 'NOX', 'RM') have large regression coefficients and thus a stronger relationship with 'MEDV'. NOX, or the nitric oxides concentration, was negatively related to MEDV such that more nitric oxides relates to a lower median home value. Conversely, CHAS and RM were positively related to MEDV, such that homes on the river and homes with more rooms were related to a higher median home value. To build off this baseline model, we ran several additional models to evaluate which home features best predict home values in Boston. These models include: lasso, lasso with interaction terms, PCA regression, single decision tree, random forest, and gradient boosting.

Regularization - Lasso Regression

Due to high correlations seen between many variables in our exploratory data analysis and our goal to select a subset of variables that are important in predicting home values, we chose to conduct a lasso regression. This regularization method uses shrinkage to push some coefficients towards zero thereby eliminating them from the model, increasing the interpretability of the model, and hopefully increasing the prediction accuracy of the model. We fit a well-tuned lasso regression using 'Lasso' from sklearn. We selected the hyperparameter (alpha) that resulted in the smallest test error, and calculated test MSE, train MSE, and R-squared values to assess model performance. The coefficients from the baseline linear regression model and the lasso regression model are shown in the plot below.

From this plot, we see that some predictors that were deemed 'unimportant' by the model (i.e., predictors with coefficients that were pushed to zero), and some were designated as 'important' (i.e., predictors with coefficients > 0). The following predictors were deemed important: CRIM, ZN, RM, AGE, DIS, RAD, TAX, PTRATIO, B,



LSTAT, while INDUS CHAS, and NOX were deemed unimportant. Results from this lasso regression suggest that, given our goal to determine feature importance in predicting median home values, we can eliminate these three variables from our model.

Regularization - Lasso Regression with Interaction Terms

To further explore how our predictors relate to our outcome variable, we examined whether the interaction between each of our predictors relates to MEDV. Interactions are important to explore and include in a model when the effect of one variable changes depending on the value of another variable. Lasso regression is a well-suited method to explore interactions, as we can calculate the interactions between all pairs of variables, and the model will eliminate interaction terms that do not predict our outcome variable, resulting in a model that is still interpretable. We selected the ten predictors that were deemed 'important' by our lasso model and created interaction terms between each pair of variables. We conducted a well-tuned lasso regression (as described above) with the important predictors and their interactions on the training sample, and evaluated the model performance with train and test MSE and R-squared values (see summary table below).

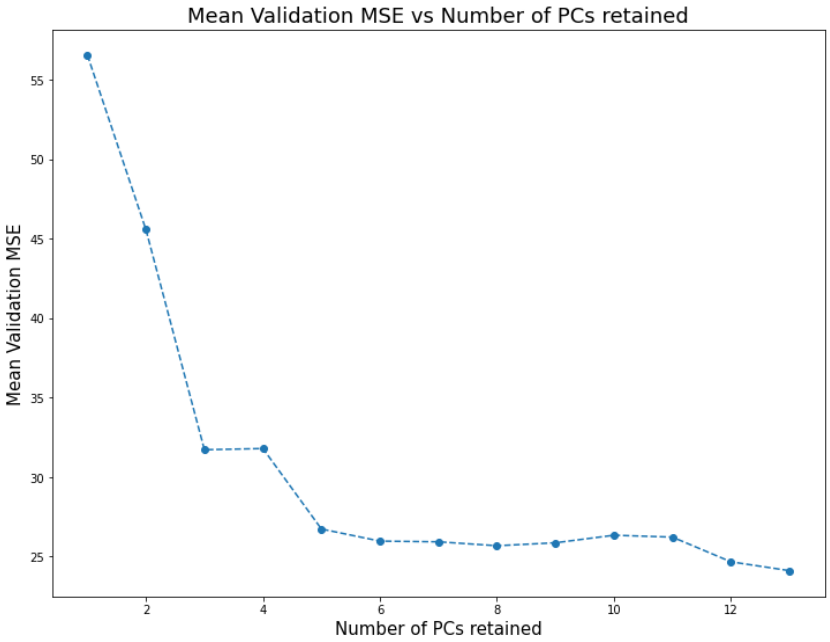
	Model	Training MSE	Test MSE	R-Squared
0	Linear Regression	21.404165	23.111529	0.684845
1	Lasso Regression	23.341191	22.760518	0.689631
2	Lasso Regression with Interaction Terms	21.873995	23.239585	0.683099

As expected, many of the coefficients of the predictors were shrunk to zero, however, the following predictors remained 'important' (two variable names indicates an interaction term): CRIM, RM, DIS, RAD, TAX, PTRATIO, ZN B, RM B, AGE B, TAX B, and TAX LSTAT. We see that lower crime rates, more rooms, closer distance to Boston employment center, more accessibility to highways, a higher tax rate, and a lower pupil-teacher ratio predict higher median home values in Boston. From the interaction terms, we see that several variables interact with B, which refers to a formula that includes the number of black individuals. This formula makes this variable hard to interpret, however, as larger proportions of black individuals yield lower numbers up until about 0.6, where larger proportions of black individuals yield higher numbers. (The documentation indicates this may be because a parabolic relationship is expected between the proportion of black individuals and housing prices.) To fully interpret the interaction terms, we would need the raw data to look at the raw proportion of black individuals (without the transformation based on the hypothesized relationship with home value).

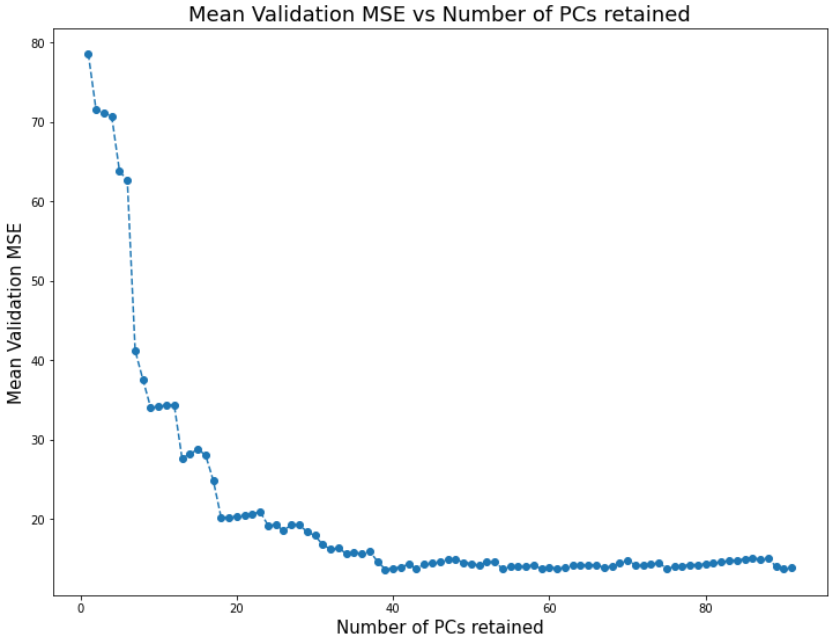
PCA Model - Controlling for Collinearity

As mentioned earlier, we noticed that some of the predictors had rather high correlations during EDA. For instance, DIS is highly correlated with INDUS, NOX, and AGE. In addition, INDUS is highly correlated with NOX and TAX. Multicollinearity may explain why our baseline linear regression model does not generalize well. So we performed a Principal Component Analysis to see whether these issues could be alleviated and lead to a better result (and perhaps fewer predictors). As before, we split the full dataset into an 80% / 20% train-to-test ratio and scaled using scikit-learn's StandardScaler. We used PCA transformation along with a 10-fold cross validation linear regressor for components ranging from 1 to 13 (the total number of predictors). To find the optimal number of components, we plotted the mean MSE of each iteration in the figure below.

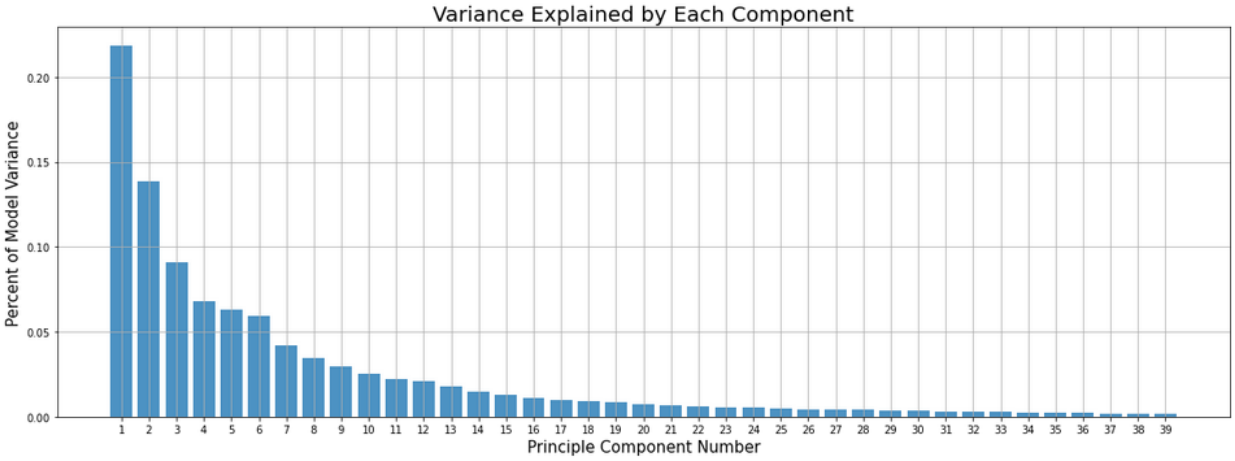
One can see from the plot that 13 is the optimal number of principal components. Because the original model has 13 predictictors, this means the optimal PCA transformation did not perform any better than the baseline.

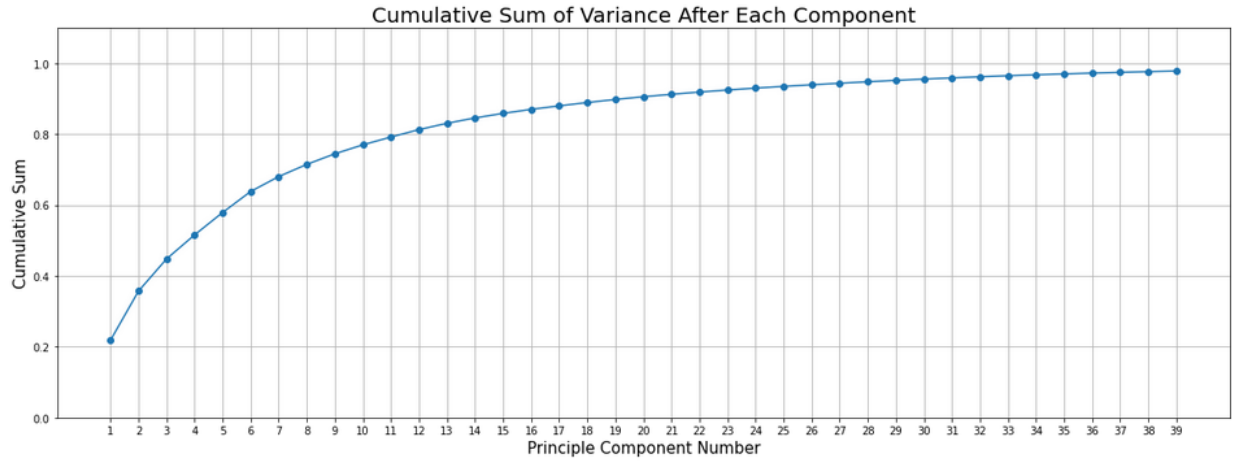


Another approach to alleviate multicollinearity is to include interaction terms and then perform another Principal Component Analysis. The idea is that interaction terms between highly correlated variables may be reduced in a PCA transformation, thereby leaving a stronger model. To do this, we employed scikit-learn's PolynomialFeatures to create only interaction terms between the 13 original predictors. The action results in a total of 91 predictors for the PCA transformation. As before, we used a 10-fold cross validation for components along the entire range of predictors (from 1 to 91). A plot of the mean MSE is found below.



Although not easily seen in the plot, the optimal number of principal components is 39 with an MSE of 9.697. When this PCA transformation is applied to the test data and fitted in a linear regression model, the resulting MSE is 12.049 and R-squared is 83.58%--a significant improvement over the baseline. By eliminating all of the principal components corresponding to small variances, the interaction PCA model reduces the prominence of the multicollinearity issues observed during the EDA phase. The figures below show the amount of model variance explained by each of the 39 principle components.

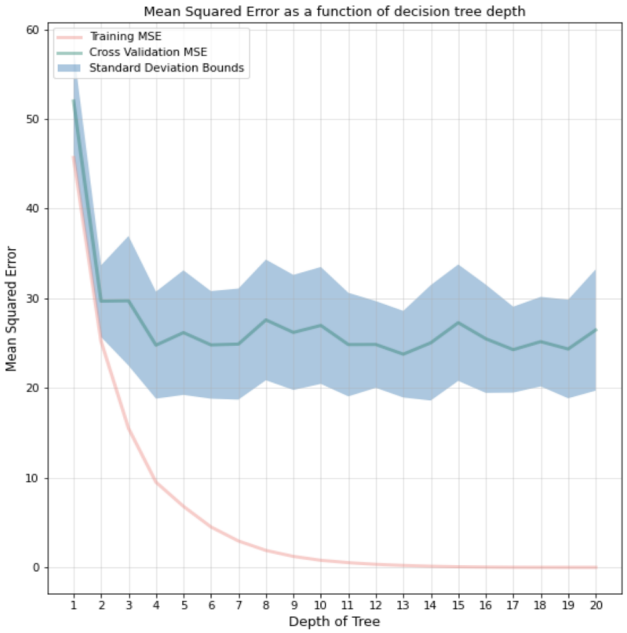


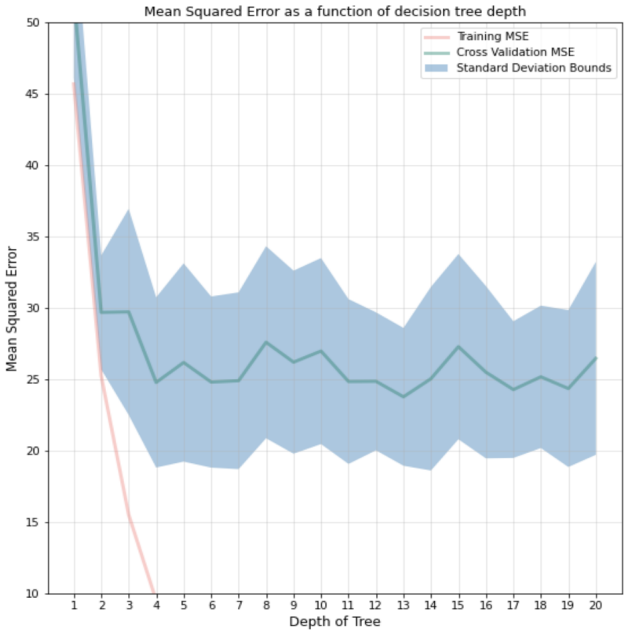


One tradeoff with the interaction PCA model is the loss of interpretability it suffers. Although it does perform substantially better than linear and lasso regression, it is not easy to explain the relationship between model outcomes and the original predictors. Afterall, each PCA component is a linear combination of the original predictors and the 78 interaction terms. Teasing out these relationships may be more effort than it's worth, especially considering that we are trying to better understand how the predictors affect median home value. Ultimately there are other methods we can use to create models that are more straightforward with possibly better performance.

Decision Tree Regressor

Moving forward in our analysis, we assessed the performance of several ensemble methods on the data, starting with a Decision Tree Regressor. To start off, we wanted to find a single tree that performed the best on the data we had by tuning the 'max_depth' hyperparameter. Using cross validation, we determined that the best decision tree had a depth of 13, as this depth minimized the value of the validation mean squared error, shown in the plots below.





Bagging with Decision Tree Regressors

Next, we wanted to compare the performance of this single tree to a bagging model that uses overfitting decision tree regressors as weak learners. To do this, we first created 55 bootstrapped sets of decision tree predictions for both the training and test data, and then computed the average of all of the predictions to obtain the bagging model's predictions on the training and test data. As expected, we found that the bagging model performed better than the single tree chosen by cross validation, as evidenced by the test MSE for the former being lower than the latter.

Random Forest Regressor

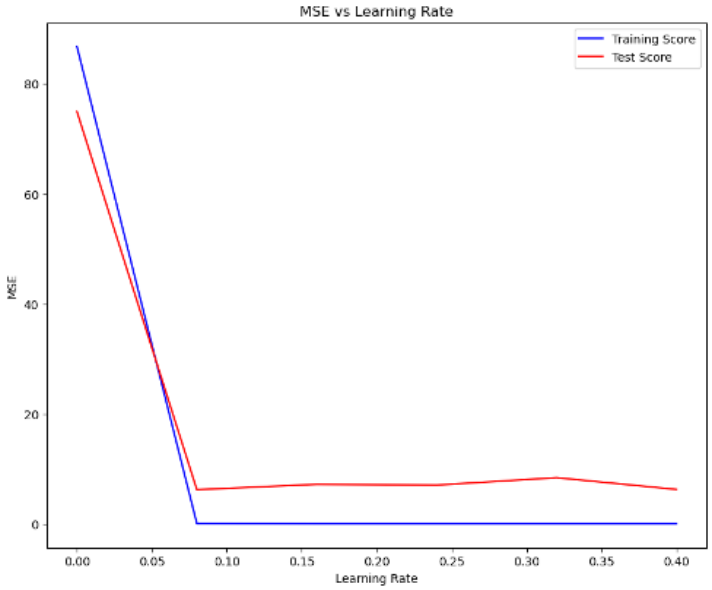
Subsequently, we fitted the data on a random forest regressor to compare its performance to that of the bagging model. Initially, we expected that random forest would perform better than bagging because random forest has many more predictors as the top node of the decision tree than bagging. Our initial hypothesis was that because random forest compares the importance of other predictors besides the particularly strong predictors, this implies that random forest will have a lower test MSE than bagging. In bagging, a predictor that's especially strong will usually dominate the first split, which could make bagging have a higher test MSE than random forest. However, our test MSE values for random forest and bagging showed that bagging performed slightly better than random forest. We believe that this is due to the fact that we only do one split of the data set into the train and test set, which could result in the test set being skewed at random. This could have resulted in the random forest model overfitting more to the noise in the test set, compared to the bagging model.

Gradient Boosted Machine

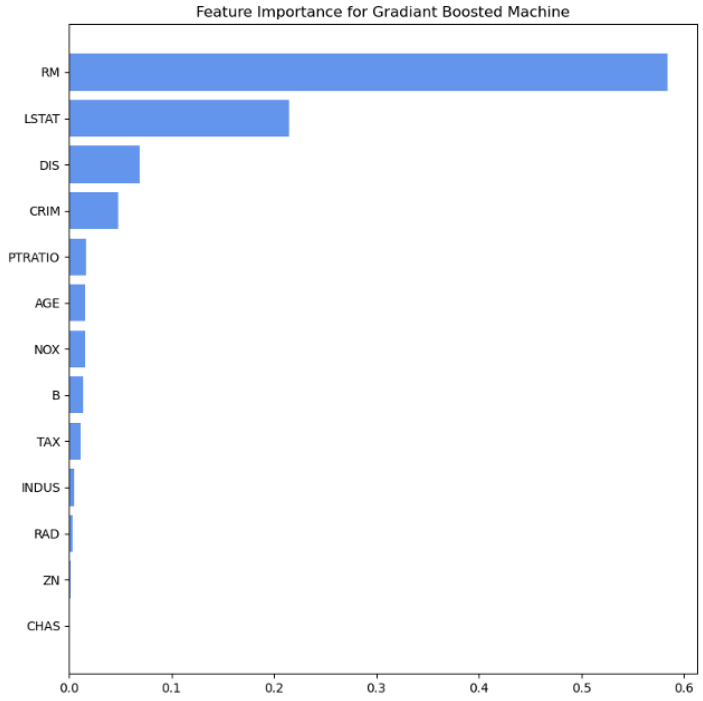
The next ensemble learning algorithm we applied to our dataset was a Gradient Boosted Regressor, which converts weak learners into strong learners. In boosting, each new tree is modified. Like AdaBoost, in Gradient Boosting, we begin by training a decision tree in which each observation is assigned an equal weight. After evaluating our model, a higher weight is put on observations that the model failed to predict accurately. The next tree is then weighted on this data. Then, the third tree is trained to predict the new residuals. This process is repeated for a given number of iterations. The final prediction is a weighted sum of the predictions made by all the trees. A unique feature of Gradient Boosting is that it uses a different algorithm to identify the shortcomings of the weak learners. This method uses gradients to allow optimization of a differentiable loss function, which measures how well the model's coefficients are fitting the underlying data.

In our initial Gradient Boosted Regressor, we initialized a model with 800 estimators, each tree had a maximum depth of 3, a learning rate of 0.08, and used a loss function We trained our model using the thirteen covariates to predict our response variable, MEDV, which is the median value of owner-occupied homes in 1000's of dollars. The model obtained a train MSE, test MSE, and r-squared of 0.0400, 6.007, and 0.9182, respectively.

An issue with GBRs is that they learn quickly, which can result in overfitting of our training data. One way we can avoid overfitting is to reduce the learning rate. We tested six different gradient boosting machines with varying learning rates from 0.00001 to 0.4. From the plot to the right, we observe that the training MSE is zero once the learning rate is 0.08 or greater. The model seems to be overfitting here. Thus, we continued to use a learning rate of 0.08 for further analysis.



We were also interested in seeing how the depth of our tree changed the MSE over each iteration. First we plotted the number of iterations against the MSE for the training and test score for trees that had depth 1, 3, 15, 20. We found that for the trees with higher depth, the MSE for the training approached zero at fewer iterations than seen in trees with depth 1 and 3. This indicated that the model was using overfit trees in its boosting. The test MSE also never goes below 10 in gradient boosters which use depths 1, 15, and 20. Thus, we felt confident to continue with a gradient boosting machine with depth 3 and learning rate 0.08.



We then proceeded to get the feature importance of the model, shown in the plot to the left. The features which were most important in predicting MEDV (in order from greatest to least) were RM, LSTAT, DIS, CRIM, PTRATIO, AGE, NOX, B, TAX, INDUS, RAD, ZN, CHAS. RM is the average number of rooms per dwelling and LSTAT was the percent of lower status of the population. It makes sense that these predictors were most important in predicting MEDV. They are also the predictors with the highest coefficient values in the linear regression models. The bagging and random forest models also used these predictors as the top predictors in their ensemble trees most frequently. ZN was the proportion of residential land zoned for lots over 25,000 square feet and CHAS indicated if the tract bounds the Charles River or not. These features were deemed less important in our gradient boosted machine.

In comparison to the other four trees we used, gradient boosting performed the best. It had the lowest test MSE and the highest R-squared between the predicted and observed values for MEDV. Because we weighted the misclassified samples in each iteration, it makes sense that this model outperformed the others.

	Model	Training MSE	Test MSE	R-Squared
0	single depth-12 tree chosen by CV	0.039230	11.340924	0.845352
1	single overfit depth-20 tree	0.000000	23.594706	0.678256
2	bagging 55 depth-20 trees	2.306482	8.602541	0.856180
3	random forest of 55 depth-20 trees	1.811040	11.360645	0.845083
4	gradient boosting machine	0.040035	6.100708	0.916809

Support Vector Machine

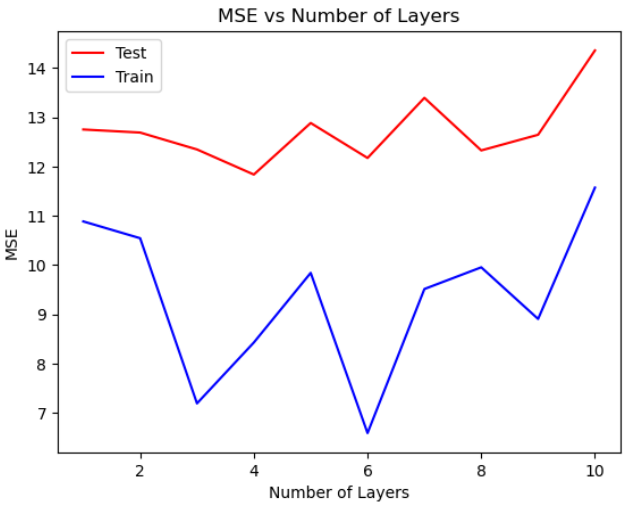
Support Vector Machine is a kind of regression method where the straight line that is needed to fit the data is referred to as a hyperplane. The hyperplanes are decision boundaries that are used to predict continuous output. The support vectors are data points on either side of the hyperplane, and which are closest to the hyperplane. Another hyperparameter used in SVM is the kernel, which is a set of functions and transforms it to help find a hyperplane.

We used two types of kernels: radial basis function and linear. The SVM with the radial basis function had a train MSE of 28.9, a test MSE of 27.19, and a R-squared of 0.63. The linear SVM had a train MSE of 24.3, a test MSE of 28.92, and a R-squared of 0.61. This model did worse than the others which was surprising. However, SVMs do have several disadvantages which could have caused this. They do not perform well when the data has more noise and target classes are overlapping. Additionally, there are

y-values which overlap and fall into similar ranges, so this may be why our model performed worse than most of the other models.

Multilayer Perceptron Regression

MLP regression is a supervised learning algorithm that learns a function which goes from R^M R^0 where m is the number of dimensions in the input and the output (13) is the number of dimensions for the output (1). Between the input and output, there are hidden layers. Each neuron in each hidden layer between the input and output layers takes the sum of their weighted inputs. The output layer receives the values from the last hidden layer. MLP regression also uses backpropagation which uses the square error as the loss function.



We created an MLP with one layer with 100 hidden units. This model had a training MSE of 10.88, test MSE of 12.749, and an R-squared of 0.82. We wanted to see if adding more layers to our model would improve the predictions and lower the MSE. We tried having between 1 and 10 layers with 100 units and calculated the MSE. There was no direct correlation between the number of layers and MSE. Our test MSE surprisingly got worse with more than 4 layers. Our lowest test MSE was at 4 layers. This model had a training MSE of 8.42, test MSE of 11.835, and an r-squared of 0.838.

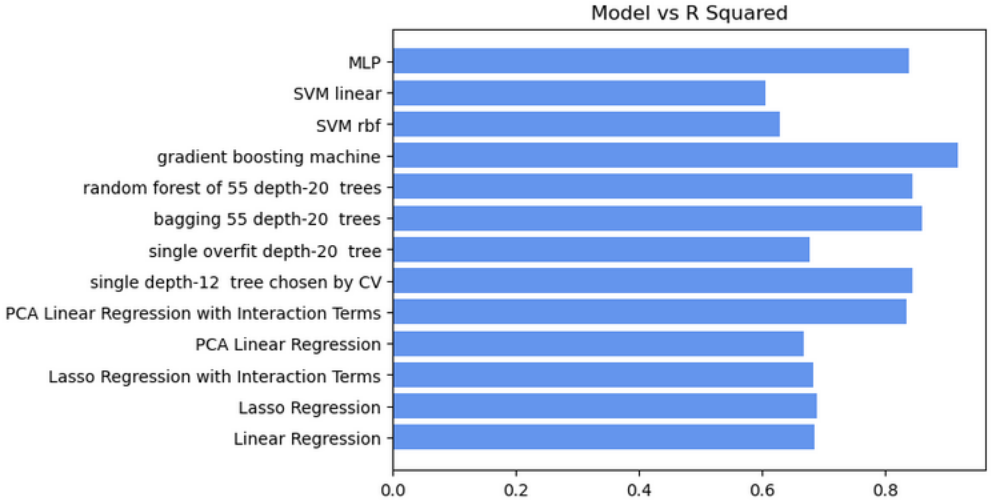
Comparison of Models

The final training MSE, test MSE, and R-squared for each model is displayed in the table. The model which performed the best was the gradient boosted machine. The model which performed the worst was the linear SVM.

Among the linear regression models, the PCA regression with the interaction terms performed the best with a test MSE of 12.04 and R-squared of 0.84. This model performed better than both SVM models. The best tree was the gradient boosted machine which had a test MSE of 6.02 and an R-squared of 0.91. The multi-layer perceptron had a test MSE of 11.84, and an

	Model	Training MSE	Test MSE	R-Squared
0	Linear Regression	21.404165	23.111529	0.684845
1	Lasso Regression	23.341191	22.760518	0.689631
2	Lasso Regression with Interaction Terms	21.873995	23.239585	0.683099
0	PCA Linear Regression	21.641378	24.290650	0.668766
1	PCA Linear Regression with Interaction Terms	9.696547	12.038558	0.835839
0	single depth-12 tree chosen by CV	0.039230	11.340924	0.845352
1	single overfit depth-20 tree	0.000000	23.594706	0.678256
2	bagging 55 depth-20 trees	2.114732	8.590848	0.861334
3	random forest of 55 depth-20 trees	1.811040	11.360645	0.845083
4	gradient boosting machine	0.040035	6.021007	0.917896
0	SVM rbf	28.937015	27.195258	0.629158
1	SVM linear	24.636993	28.920805	0.605628
0	MLP	8.428066	11.835810	0.838604

R-squared of 0.84. This performed better than the SVM models, the overfit single decision tree, PCA linear regression, the lasso regression, and the linear regression. The charts below display the tabular data in a more digestible format, and one can easily see that the gradient boosting machine had both the smallest MSE and highest R-squared of all the models created.



Aside from performance, another important factor for determining the best model is interpretability. The overarching goal of our project was to find out which predictors or combination of predictors had the greatest effect on MEDV, the median home values. Overall, the gradient boosting machine had the best performance and interpretability for understanding these relationships. Using this knowledge and a well-performing model, we hope to explore its applicability in predicting rates of homelessness. However, more research would be needed to find an appropriate and timely dataset to accomplish this next step. We believe that our project has been a good start to achieving this goal, and also helped solidify many of the concepts we have learned throughout the semester.