1) Statistical Analysis and Data Exploration

- Number of data points (houses)? **506**
- Number of features? **13**
- Minimum and maximum housing prices? **min = 5.0 , max=50.0**
- Mean and median Boston housing prices? **mean = 22.5328, median = 21.2**
- Standard deviation? **9.188011**

2) Evaluating Model Performance

- Which measure of model performance is best to use for predicting Boston housing data and analyzing the errors? Why do you think this measurement most appropriate? Why might the other measurements not be appropriate here?

Since price of a house is a continuous data and not discrete, this becomes a regression problem. The measurement model is to look at deviation of predicted price from the actual price of the house and not make a binary classification. Hence measurement of Accuracy, Precision, Recall and Fscore don't apply. Mean squared error or Mean absolute error are most appropriate.

- Why is it important to split the Boston housing data into training and testing data? What happens if you do not doest this?

Machine learning is about inductive reasoning. We use models on existing datasets to generalize a function that would predict for new data. Splitting data into training and testing is important to be able to verify if our model suffers from underfitting or overfitting errors. Analysing training errors with our model helps in understanding if our models captures all needed metrics in generalizing a function to predict the output (i.e. underfitting errors) and Analysing testing errors help in catching overfitting errors, such as model having very low errors with training data but higher errors for the test data.

- What does grid search do and why might you want to use it?

Grid search automates the process of guess and check iterative approach used to find the optimal parameters for any given training model while working with data. Every training model has various parameters that could be used to smoothen or increase sharpness of its estimates. In case of GausianNB, the value of k (laplace - smoothing) is used to reduce sharpness in prediction of posterior probability. In Boston Housing we use DecissionTreeRegressor which I don't fully understand currently, but the max_depth parameter in it tunes the sharpness of estimates. The manual way would be to run Kfold cross_validation for each different max_depth parameters and analyse the error score for each to figure out the right max_depth. Grid search automates this process by taking in range of max_depth to iterate on and selects the best estimator with the ideal max_depth value which minimized variance and bias.

- Why is cross validation useful and why might we use it with grid search?

Splitting data into training and testing is not as simple as it seems. If we were to pick the first half in a 50:50 split, we could run into dangers where first half of data is only of one type and the second half is of other types. Hence model trained on first half will perform poorly on the second half data. Also selecting the right sample to test or train is not an easy problem. The sample has to be contain all possible data dimensions for model to be trained and tested correctly. Cross validation helps over here, cross validation helps to split the data into train and test samples and can take care to shuffle the data. Also KFold cross validation enables efficiently using all the data in both testing and training.

3) Analyzing Model Performance

- Look at all learning curve graphs provided. What is the general trend of training and testing error as training size increases?

In case of the Training curve for lower training size the the training error is low and as we increase the training size the training error increases but it approaches a constant value. As training size increase this constant error can be attributed to bias/underfitting of our model. We can notice this constant value decreasing as we increase model complexity in case DecisionTreeRegressor that would max_depth parameter.

In case of Testing curve, we can see that as training size increase the testing errors comes down, this is because when we test our model post training on a small dataset size, The model is not trained to all types of data in the dataset and hence suffers from variance/overfitting it needs more data to generalise. As training size increases the model is trained to all types of data and hence generalises better to the data and hence reduces the test errors.

- Look at the learning curves for the decision tree regressor with max depth 1 and 10 (first and last learning curve graphs). When the model is fully trained does it suffer from either high bias/underfitting or high variance/overfitting?

The learning curve graph with max_depth = 1, clearly suffers from bias/underfitting as the training error itself is qutie high. The graph with max_depth=10 suffers from high variance/overfitting as here there is big gap between the training error line plot and test error line plot for different test instance sizes.

- Look at the model complexity graph. How do the training and test error relate to increasing model complexity? Based on this relationship, which model (max depth) best generalizes the dataset and why?

From the model complexity graph we can see that training error decreases and tends to zero as model complexity increases but the test error line plot reduces till max depth of 3 post

which it waves around the same error value for increase in max_depth and then the line tends to increase in error towards end. I feel the max_depth around 3.7 best generalizes the data set and as max_depth should be an integer max_depth of 4 would be the best. Also if we run the boston_housing.py GridSearchCV implementation to estimate the best_params_ we can see than on multiple runs the most recurring value (median) would be max_depth=4 as the best fit.

This is the point post which the difference between the test and training errors increase and post which test error doesn't decrease further. max_depth beyond this just increases complexity leading to overfitting.

4) Model Prediction

- Model makes predicted housing price with detailed model parameters (max depth) reported using grid search. Note due to the small randomization of the code it is recommended to run the program several times to identify the most common/reasonable price/model complexity.

On multiple runs the best_estimator selected by GridSearch was most of the times with max_depth = 4. This relates to my understanding of the learning curve and model complexity graphs as being the right parameter with low bias and variance.The predicted price of the house for max_depth = 4 is 21.62974359

- Compare prediction to earlier statistics and make a case if you think it is a valid model.

The result of predicted price of 21.629 lies close to median price calculated for the given dataset which is 21.2 and mean price being 22.56, This prediction seems reasonable as while observing scatter plots relationship between each attribute and target price certain attributes such as RM (Rooms per dwelling) seemed to linearly affect the price of the house with an

upward slope while certain attributes such as NOX had and downward slope. Some such as CRIM allowed high variance in price of houses for smaller CRIM values and reduced price on higher CRIM. Comparing these attributes against this house attributes shows that this house would lie closer to the mean/median of these house prices.