## Data Exploration -
Statistical Analysis

```
Size of data?  506
Number of features?  13
Minimum Value  5.0
Maximum Value  50.0
Calculate Mean  22.5328063241
Calculate median  21.2
Calculate standard deviation  9.18801154528
```

## Evaluating Model Performance

**Performance Metric:**
I am evaluating performance of model using mean_squared_error. This metric penalizes large differences between predicted values and true values.

**Testing/Training Split:**
I have split data set into **Testing/Training** to evaluate the model on the data it has not seen. I am holding out 40% for testing and retaining 60% for training.
I am using cross_validation train_test_split to separate the data set into Training features and labels and test features and labels

**Cross Validation & Gridsearch**

I am using 3 fold **Cross validation** to use 3 separate train/test sets that are randomly generated multiples times in order to evaluate the algorithm at each fold. Score of each test is then averaged to get the average score of the model

Grid search is used to find and evaluate model for best tuning parameters. Testing against 10 depth parameters grid search finds the best parameter optimizing training and test errors .

## Analyzing Model Performance

### Learning Curves and Training Analysis

**After fitting a model with 60% of training data and testing with 40% data of data I tested `Decision Tree with Max Depth from 1 to 10.`**

### Learning Curves and Bias & Variance Analysis

To examine relationship between training and testing errors we split training data in 50 folds and use them to train our model (Decision Tree Regressor) on each commutative fold measuring Median Square Error.

In order words in our 60/40 split we train on 303 examples. We split 303 examples in 50 folds, with each fold containing approximately 6 examples.
In the first fold we train on 1, second on 7 third on 13, examples.

**What is the graph trend of training and testing error as training size increases?**
As one can imagine the more examples we train on the lower the error rates are for both training and test errors

We can see that as the training error decrease, test error does the same, however test error can never be lower than the training error.
As the training sizes increases and as the complexity of the model incenses training error dirops to zero, while after tree death of 5 the test error doesn't improve. This indicates that higher depth models fit data too closely and don't generalize well.

```
Decision Tree with Max Depth:  1
With Max Depth of 1 we can see that model is not predictive. It has high test and training error thia is due to lack of generalization and learning as the
decision tree can only split on one level,
```


```
Decision Tree with Max Depth:
2
```


```
Decision Tree with Max Depth:
3
```


```
Decision Tree with Max Depth:
4
```

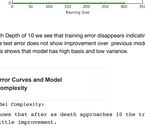

```
Decision Tree with Max Depth:
5
```


```
Decision Tree with Max Depth:
6
```

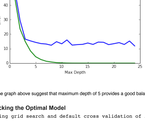

```
Decision Tree with Max Depth:
7
```


```
Decision Tree with Max Depth:
8
```


```
Decision Tree with Max Depth:
9
```



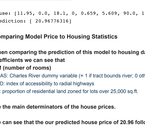

```
Decision Tree with Max Depth:
10
```




With Depth of 10 we see that training error disappears indicating that the model has memorized the data.
The test error does not show improvement over previous models with depth higher than 5.
This shows that model has high basis and low variance.

### Error Curves and Model Complexity

```
Model Complexity:
Shows that after as death approaches 10 the training error disappears while test error shows
little improvement.
```




The graph above suggest that maximum depth of 5 provides a good balance between bias and variance

### Picking the Optimal Model

Using grid search and default cross validation of 3 we examine the affects of parameters on the model while re-fitting with each irritation.

```
Final Model:
GridSearchCV(cv=3, error_score='raise',
       estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, random_state=0, splitter='best'),
       fit_params={}, iid=True, loss_func=None, n_jobs=1,
       param_grid={'max_depth': (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)},
       pre_dispatch='2*n_jobs', refit=True, score_func=None, scoring=None,
       verbose=0)
```

**`We find Best parameter {'max_depth': 5}`**

### Predicted Housing Price

```
House: [11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13]
Prediction: [ 20.96776316]
```

**Comparing Model Price to Housing Statistics**

**When comparing the prediction of this model to housing data we can see that the prediction is inline with training data set. Looking at the training data and after obtaining coefficients we can see that**
**RM (number of rooms)**
CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
RAD: index of accessibility to radial highways
ZN: proportion of residential land zoned for lots over 25,000 sq.ft.

**are the main determinators of the house prices.**

**We can see that the our predicted house price of 20.96 follows a similar hose**

| 'CRIM' | 'ZN' | 'INDUS' | 'CHAS' | 'NOX' | 'RM' | 'AGE' | 'DIS' | 'RAD' | 'TAX' | 'PTRATIO' | 'B' | 'LSTAT' | Prediction |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.00245 | 0 | 8.14 | 0 | 0.538 | 6.674 | 87.3 | 4.239 | 4 | 307 | 21 | 380.23 | 11.98 | 21 |