Boston Housing

Fernando Hernandez

December 11, 2015

1 Statistical Analysis and Data Exploration

This analysis is done on the Boston House Prices dataset.

There are 13 features to predict with:
Attribute Information (in order):

- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS proportion of non-retail business acres per town
- CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to 1940
- DIS weighted distances to five Boston employment centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per $10,000
- PTRATIO pupil-teacher ratio by town
- B $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT % lower status of the population

The target variable to be predicted is:

- MEDV Median value of owner-occupied homes in $1000's

Other summary statistics include:

```
Number of houses in data set: 506
Number of features: 13
Minimum house price in dataset: 5.0
Maximum house price in dataset: 50.0
Average house price: 22.5328063241
Median house price: 21.2
House price standard deviation: 9.18801154528
```

Here the median is about $1,300 less than the average. This suggests a positive skew in the price distribution with some high priced homes pulling the average away from the bulk of the data. (More on this later in the model analysis section)

2 Evaluating Model Performance

2.1 The Loss Function - What to choose?

Mean Squared Error (MSE) was chosen as the loss function to minimize when assessing the quality of our predictor of house price.

2.2 What's so great about MSE for our problem?

Pros

- **Our problem is a regression problem.**

Other measurements such as accuracy or f1 score are obvoiusly not used because they are classification measurements. We are trying to predict a semi-continuous variable (measures in dollars/cents) in house price.

- **Mean Squared Error penalizes bigger errors quadratically the more wrong they are.**

I would argue this is useful when predicting house prices because we want to maximize profit/ minimize cost as much as possible and getting too far from a good price is worse they further we get.

Mean absolute error is a possible alternative, but was not used since I felt being off from a good price gets worse the further you are, and MSE penalizes this better.

This is in part due to the nature of the data. If we were using our model to predict house sell price, being off by a little from the optimal price means we lose a bit of money (or gain a bit, yay).

But being off by too much could mean we get no offers, and the house misses opportunities to sell during peak sell months (and staying on the market longer possibly negatively affecting the potential sell price.)

- **Mean Squared Error takes in account both the bias and variance of the model.**

Minimizing Mean Squared Error minimizes bias and variance simultaneously.

Our model is hoping to approximate the true relationship between our chosen model/parameters and the price of houses.

There are three sources of error preventing this.

- ϵ: irreducible error

    1. This is error/noise that is inherent in our data. This can be any myriad of other attributes of the data that make the data slightly noisy, and unable to be modeled perfectly by a given model of given complexity.

    This noise is randomly distributed with 0 mean (otherwise it would have predictive value.)

    2. This also includes the amount of inherent error/noise that we can't reduce given a specific model and complexity (bias error.)

- *bias*

    This is how well our chosen model can model the true relationship/model between the data and the price.

    If we were run our same model, N-number of times on N-number of samples from the population, we could get an average model fit.

    Bias would be how far off from this average model is from the true model (in the limit as N approaches the population size.)

    More explicitly:

$bias = model_{True} - \hat{model}_{Avg}$

where:

$\hat{model}_{Avg} = \frac{\hat{model}_{sample1} + \hat{model}_{sample2} + ... + \hat{model}_{sampleN}}{N}$

A simpler model would have higher bias because it would be, on average, further from the true model. It would be biased toward itself, regardless of the true model.

This is known as underfitting.

Models with higher complexity are much more flexible, so are able to ***on average*** ($\hat{model}_{Avg}$) fit a given function better giving them lower bias.

- *variance*

  Variance is how much each individiual model varies from the average model when given a new sample of data.

  This difference is then squared and averaged across all possible samples with our given model.

  More explicitly:

  $variance =$

  $$\frac{1}{N} \sum_{sample(i)=1}^{N} (model_{True} - \hat{model}_{sample(i)})^2$$

  A more complex model has higher variance because it has more flexibility to bend itself to a new sample data each time it sees new data. It would be able to fit each new sample dataset better, but have more variation between different sample datasets, and lose some generalizability to unseen data sets.

  This is known as overfitting.

  Simpler models are less impressed with new data, and give more consistent predictions when generalizing to new data.

Mean Squared Error is $bias^2 + variance + \epsilon^2$. There is a bias-variance tradeoff, as we reduce one - in general the other will rise. Minimizing mean squared error seeks to find a balance between the minimization of both.

Cons One possible drawback though might be that Mean Squared Error heavily weighs outliers since it is an average of *squared* errors. This is can be an issue depending on our application, or the distribution of the target variable being predicted.

2.3 Training, validating, and testing

Training, validation, and testing splits are absolutely vital when trying to evaluate a model, or when picking a set of parameters in a chosen model.

We should be careful to find and tune our models using cross-validation on a validation set. Then once we are confident we have a good model and parameter settings, we should test once on another test set of unseen data.

If we don't have enough data or computational resources, K-fold cross-validation only can be a compromise. K-fold cross-validation holds out K-fold amount of data within our training set to predict on for each of it's iterations.

2.3.1 What K and why?

For this analysis, we'll use 5-fold cross validation instead of the default of 3.

If K is too low, the amount of data for training the model (66% in the case of 3-fold) would be lower not giving our model as much data to train on. This can cause higher variance between models trained

We also don't want to k-fold splits to be too high, since as K approaches N (the number of data points in our data set), we would have more and more overlap in training data used to train on. This would make all of the models more and more correlated with each other as K approaches N. This might cause another type of higher variance in our predicted models since "In general, if the variables are correlated, then the variance of their sum is the sum of their covariances".

Ideally we would want to split once around 70%/30% train/test.

Run Kfold cross-validation on a 70% training set only, then test once only on the 30% hold-out test set.

Or with enough data, further split the training set into training/validation sets, train on the training set while scoring on the hold-out validation set to adjust our model parameters.

Then once done fine-tuning, run our model once on the 30% hold-out test set to get an idea of the out-of-sample performance.

But again, this would require more data since so many splits would reduce the predictive power of our model, causing higher variance.

2.4 Pitfalls of ignoring data splitting

The biggest problem faced when not splitting the data is overfitting to the data set and having poor generalizability to new unseen data when making predictions.

If the data is not split at all using any of these methods, there can be an almost guarenteed risk of overfitting to the sample dataset.

In general, training error will continue to go down as a model becomes more complex. It will fit the model to the dataset, but have poor generalizability when predicting on new unseen data.

2.5 Grid Search to the rescue

Grid search searches over a combination of different models and/or model parameters. It looks for the best combination to minimize your loss function (or maximize a reward function.)

2.6 Verifying our search

Cross validation is important because, once again, we run the very big risk of overfitting to our particular sample dataset as we search for the very best combination to minimize loss on our model with our sample data.

2.7 Warning: Not so fast!

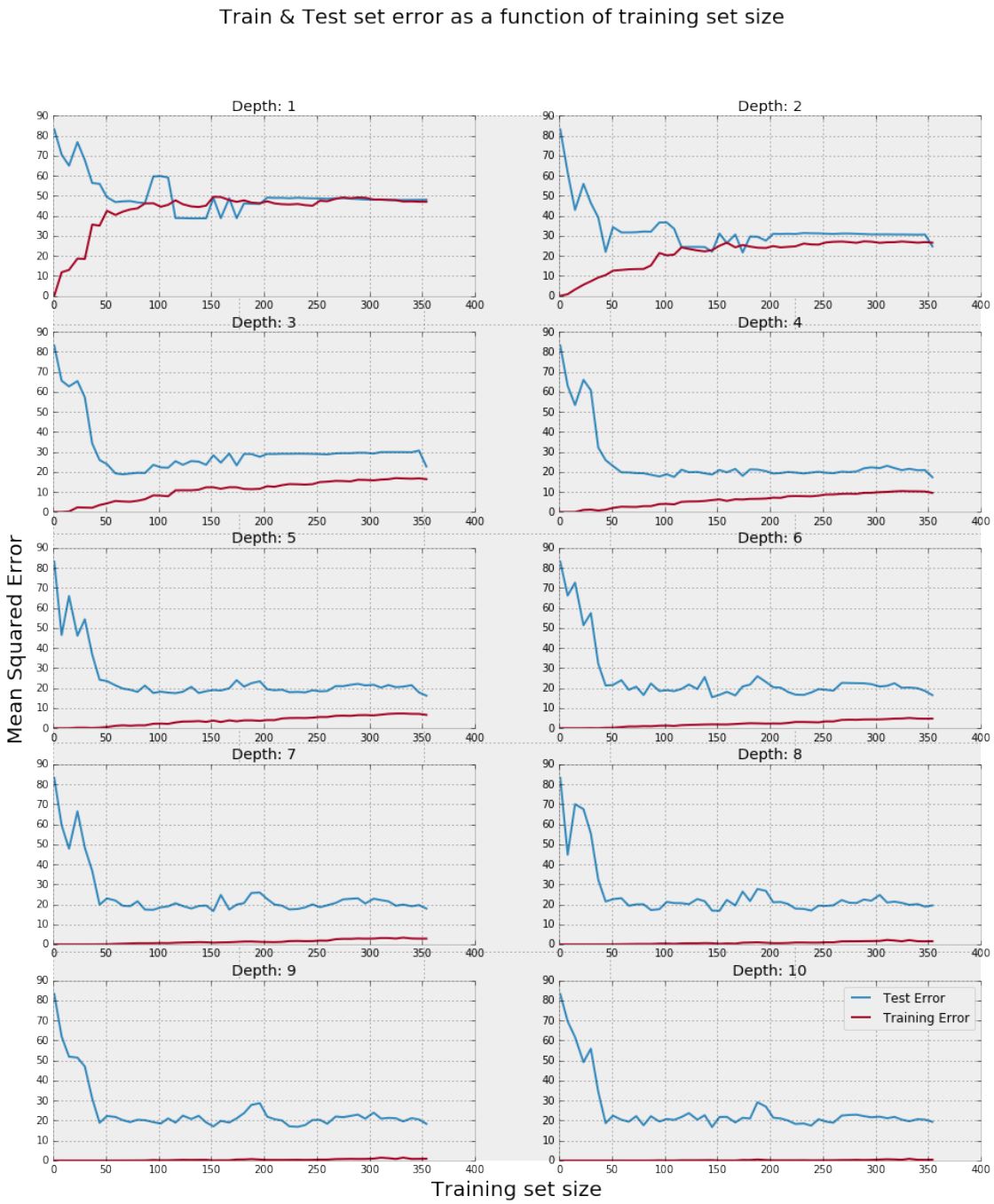
Cross-validation is not a panacea. We can overfit even when using cross-validation. If we "torture the data long enough, it will always confess".

As we try more and more combinations of parameters, ever higher levels of complexity, we get closer and closer to any one of those combinations returning a minimum loss score by random chance instead of modeling the behavior of the population data of our sample.

This is why it is important to, when we have enough data, have a validation set for cross-validation and a test to evaluate our model that cross-validation chose.

---

3 Analyzing Model Performance



3.1 General Trends with respect to Training Size

3.1.1 Test error

At a very small training set size, no good model of the data can be learned on such little data and test error is high regardless of which max depth is used.

As the training set size increases, the model has more data to learn on, and the test error drops accordingly.

3.1.2 Training Error

The training error also increases with training set size. This increase is larger for simpler model parameters (max depths closer to 1) since such a simple model cannot hope to accurately measure the complexity of the data.

As the max depth is increased, training error still increases, but much less so since it is now able to fit the the data better.

3.1.3 Training and test error convergence

Also note that as training size increases, training and test error converge. This convergence is to the limit of error ***for each given model and model complexity***.

This limit is the combination of the irreducible error from:

1. Inherent noise in the data.
2. Bias error of the given model and complexity.

But this convergent error value won't go to zero (plus noise) because even with all of data in the world, a given model with given complexity may not be flexible enough to capture the true relationships to predict price perfectly.

But in the limit of training set size as it approaches all the data possible, the test error and training error should converge to the same value.

3.1.4 Simple vs. complex

Depth 1 If we take a closer look at depths 1 and 10 for this particular data split, we can see that the training error of the simpler model converges with the test error at around a training size of 50. It has already hit the error limit (data noise + bias error.) The model can't hope to go below a Mean Squared Error of 50 no matter how much data it sees since it is too simple to model price.

This shows the model to be highly biased, regardless of new data and underfitting the data.

Depth 10 Here the more complex model has lower test error and training error, but they are no longer very close. The training error is far lower (almost zero) while the test error sits at around 20 Mean Squared Error.

The model makes predictions on the training set extremely well all the way to the full 352 samples while the out-of-sample test error is far higher. This indicates overfitting.

3.1.5 Best of both worlds

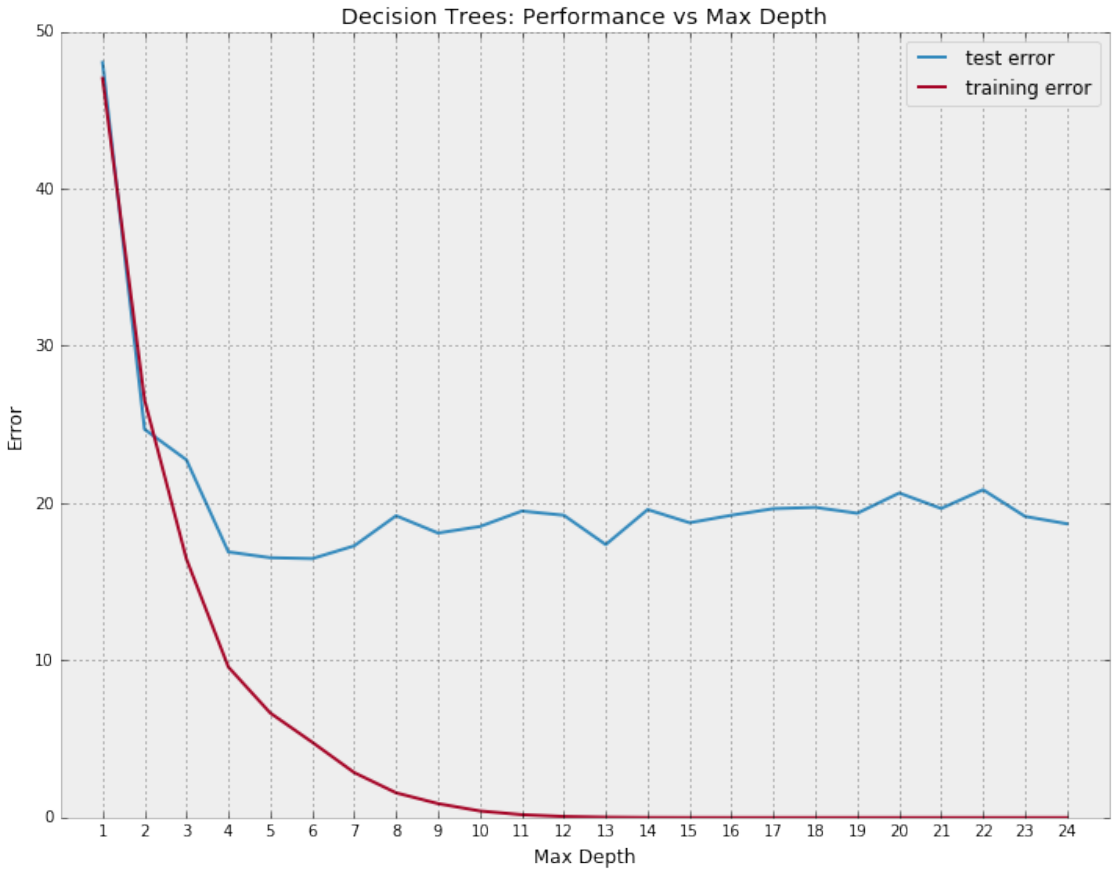
If we look further at depths 4 and 5, the training error is much closer to the test error making it less biased and better able to predict the test error. The two also converge at a bit under 20 MSE, lower than depths 1-3, while still remaining close. The error then creeps back up to around 20 at higher depths.

3.1.6 Is it really too complex?

We see that the test error is still fairly low at higher depths, but the training error doesn't quite match up. What will happen with more data? It's possible that more complexity/depth helps predict price, and with more data, the test error will drop (it's been stuck at 20 since depth 4.) But it's also possible that it's hit the limit of the noise plus model bias. It's impossible to tell without more data, and higher complexity is currently overfitting.

For now, depths 4-5 looks to be the best depths that minimize overfitting and error with our data available.

`Model Complexity:`



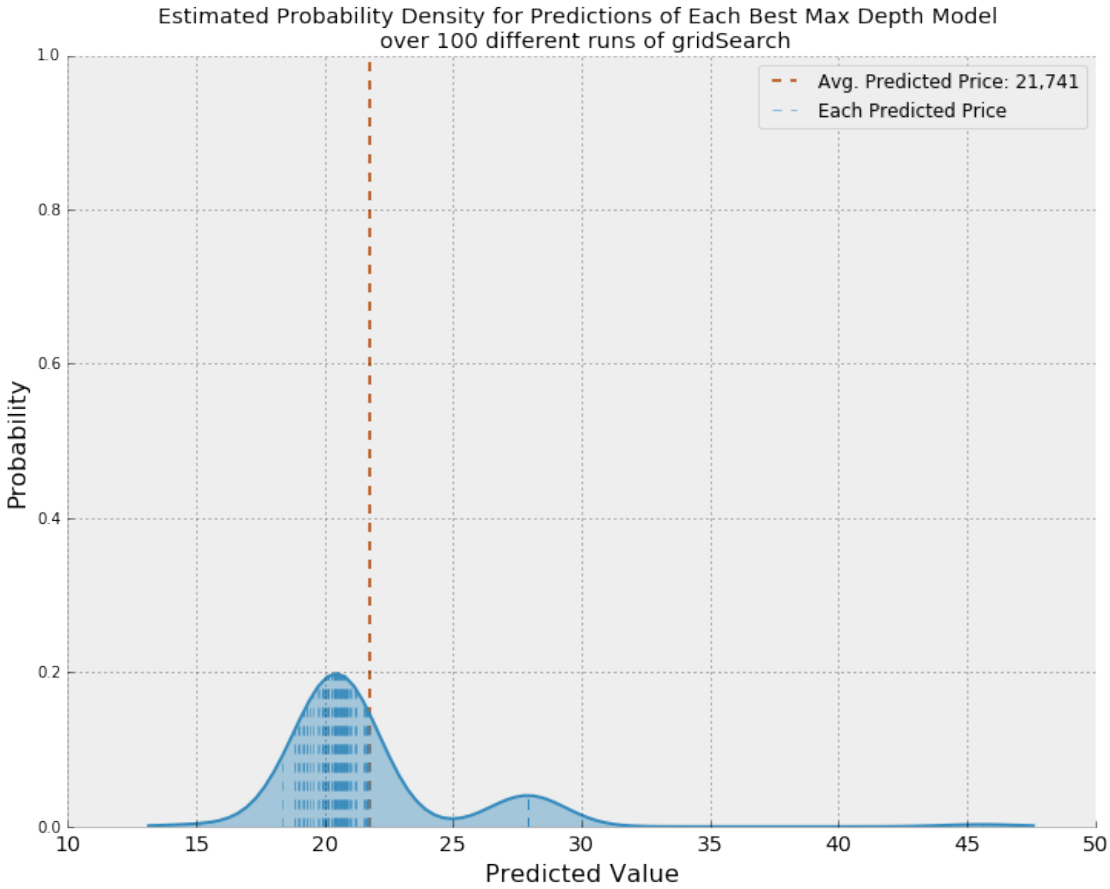
3.1.7 Model Complexity - Test and training set relationship

I would argue that test error reaches its minimum around depths 4-5. Training set error continues to drop as the model gets more complex. As a model becomes more complex, it's better able to model every sample the training set data.

But as we can see, test error plateaus, while training error tended to zero. This is the model overfitting to the training set, while not adding anything to generalizability.

---

4 Model Prediction



4.0.8 Smoothing out randomness

Due to the small data set size, randomness can occur when reporting predictions with models trained after the training/test splits.

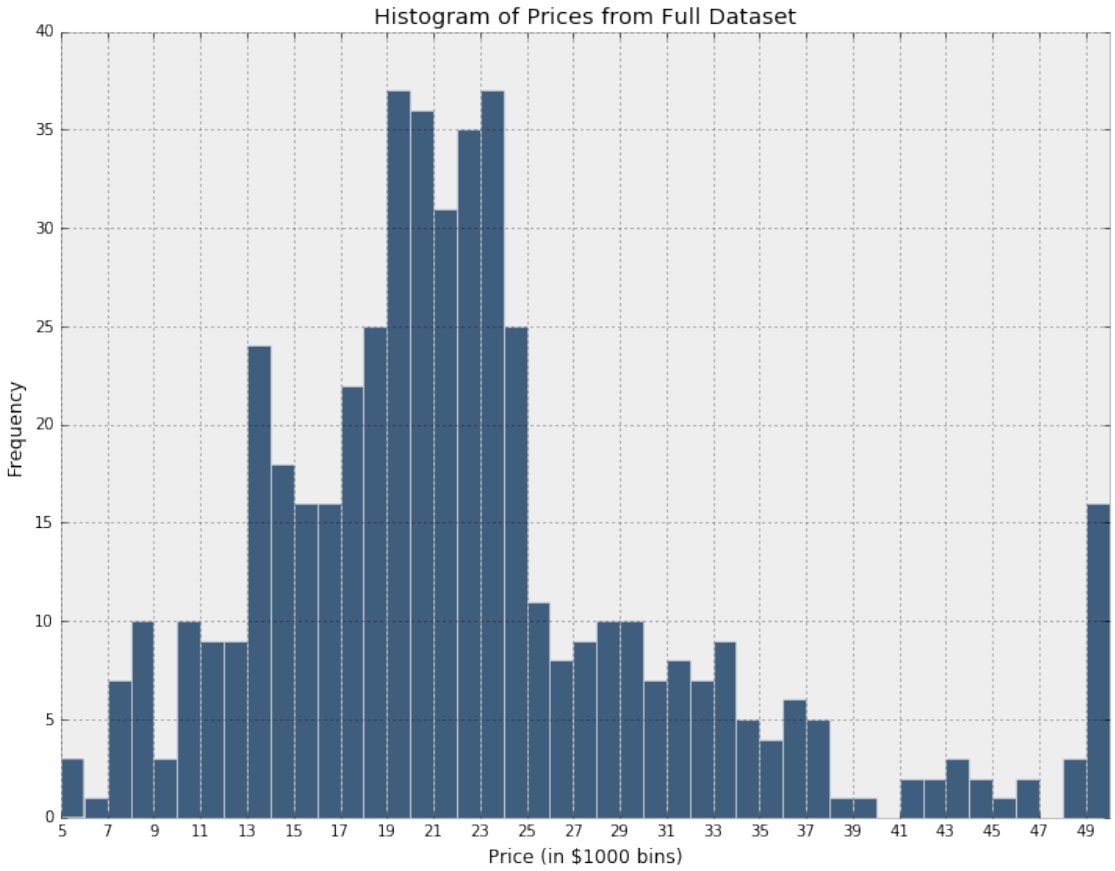
In an effort to help to alleviate this randomness, kernel density estimation was used to estimate the probability density function of the housing price based on 100 different runs with 100 predictions.

On average, over 100 different runs, the trained model predicted around $21,700 for the sample house.

In fact, we can see that the bulk of estimates and probability lie around $20,500 but at least one model predicted a very high price pulling the average with it!

In any case, this compares favorably to the average house price of $22,530, even if under-predicting the price a bit on average.

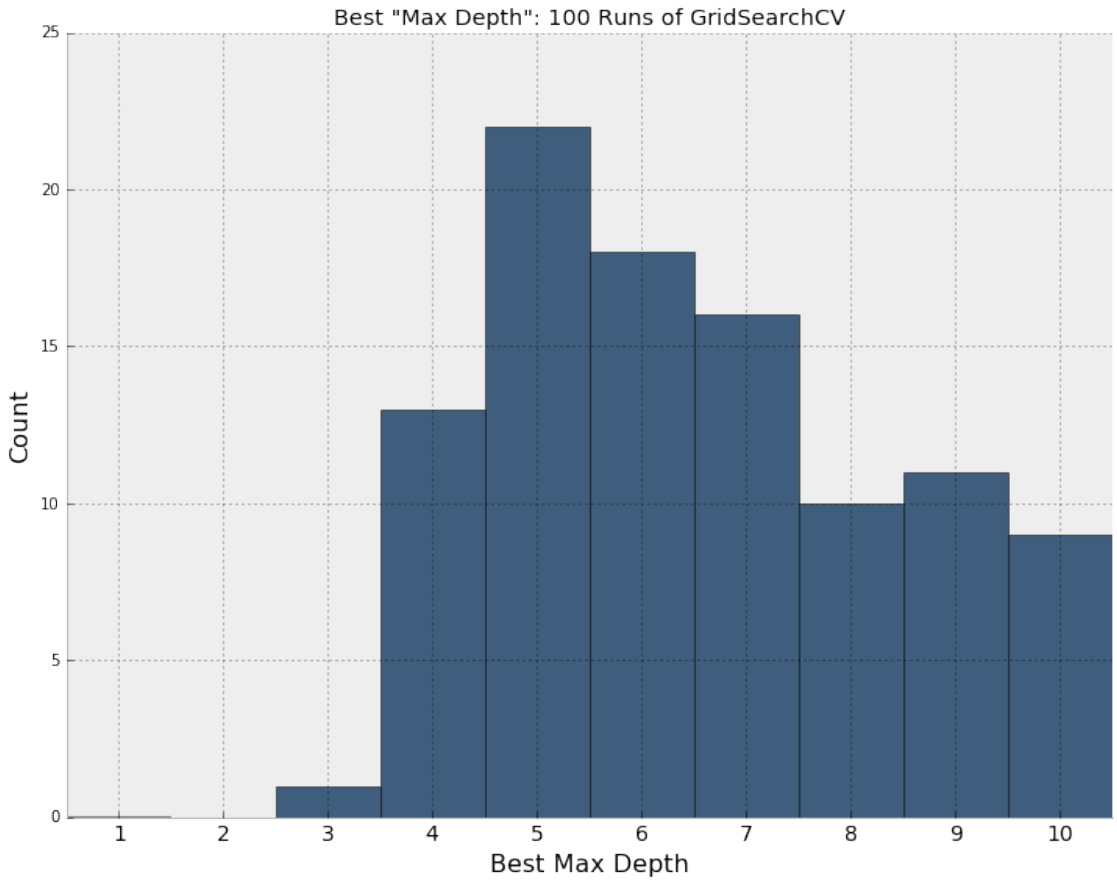
Also, the model does, on average, predict very close the median price of $21,200.



4.0.9 Compared to the true distribution

In our post-mortem analysis, if we look at the true distribution, we can see some high value prices skewing the average.

Perhaps our models can be forgiven for over-predicting heavily when given certain random splits of the data!



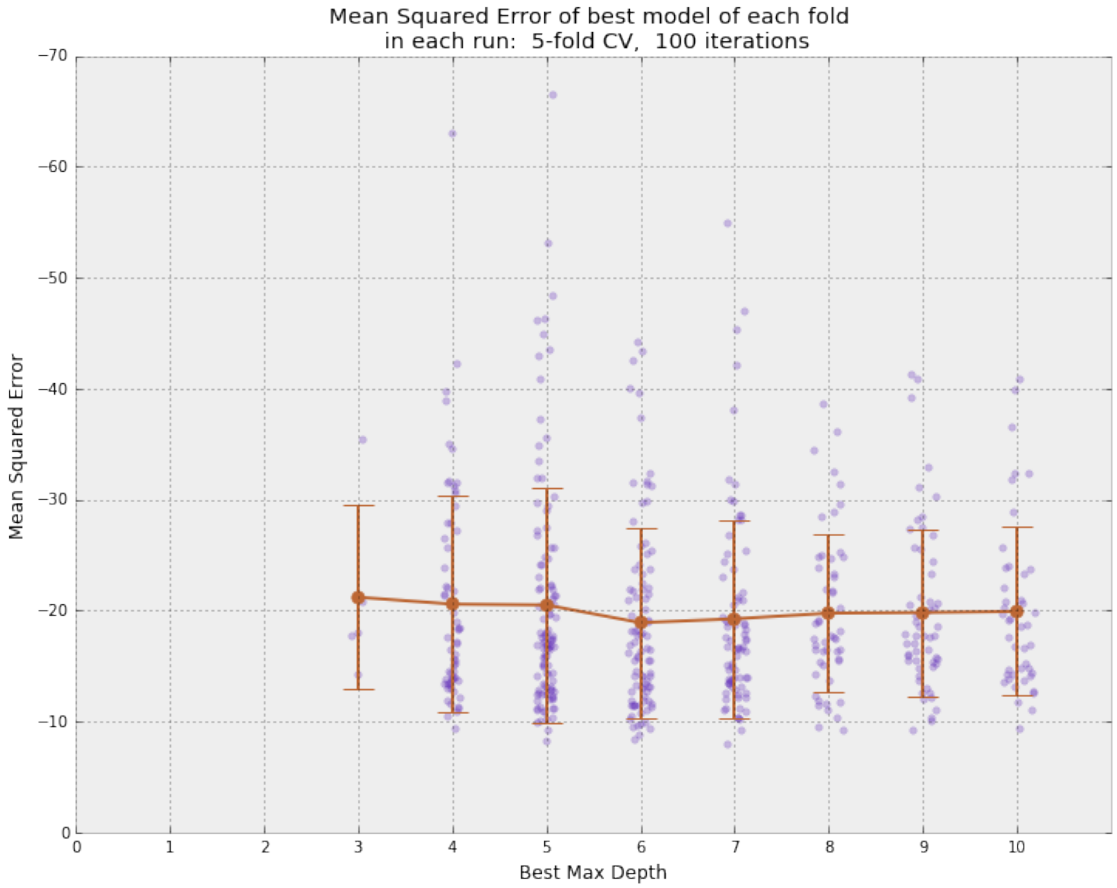
4.0.10 Best of the best

Over 100 runs of grid search with different splits of the data each time, we find that a depth of 5 is most common. The distribution is also right-skewed.

This agrees with what we saw in our model complexity plot (of one model.) There was a definite hinge at around a depth of 5.

There seems to some variability in depths chosen. This is possibly due in part to the size of the data set.

The data used was on the small side at 506. They were also split out in a 90/10 train/test split before each run to simulate real-world application. This left 455 samples sent to cross-validation. At 5-Fold cross-validation, each model was trained on 364 samples, and tested on 91 samples in each fold.



4.0.11 Spread of errors

We can see the variation of the error by looking at the spread of errors from within each fold of each best depth model.

Here there are 100 models trained, with the best model of each run picked accroding to the lowest 5-fold average error. From each model, each fold's error is plotted. So there a total of 500 MSE values (100 runs x 5 folds each) from the best models.

We also see error bars showing the standard deviation from the of each max depth's scores.

We can see that most values do fall within one standard deviation but there are a fair amount of folds where the error was much higher than usual.

Again, the smallish dataset may account for some of the variability but of course, the model itself is a fairly simple decision tree.

4.0.12 Is it any good?

Although it consistently underpredicts the true value, this seems like a reasonable model for prediction since the target variable itself has high value homes pushing the average price up.

If looking for better overall predictions, a random forest regressor might give it a little push in the right direction.
