Questions and Report Structure

1) Statistical Analysis and Data Exploration

- Number of data points (houses)?

  There are 506 houses in the dataset.

- Number of features?

  The dataset has 13 features.

- Minimum and maximum housing prices?

  The minimum house price is 5. The maximum house price is 50.

- Mean and median Boston housing prices?

  The mean and median Boston housing prices are 22.5328063241 and 21.2.

- Standard deviation?

  The standard deviation is 9.18801154528.

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?

  The mean squared error is the best measure to use.

  We are facing a regression problem and are seeking to minimize the error, i.e., the difference in predicted house price and actual house price. Therefore, we are interested in the average difference between our predictions and the actual values.

  Individual predictions could be below or above the actual house prices. We need to account for that by squaring the differences or by taking the absolute of the difference. Otherwise, positive and negative errors could cancel each other out. Therefore, our options are the following metrics: mean squared error or mean/median absolute error. If we were to use the mean/median absolute error, every error would get the same weight even if some predictions were much further off than others. We would not face this issue when using the mean squared error. By squaring each error, larger errors gain relatively more weight in the calculation of the mean error. This behavior is beneficial for us as we want to punish predictions that are very off. Why? The model could help us predict the price we could get for our house in Boston. From that perspective it would be worse if the model predicts a price that is way too high as we might then wait forever to sell our house for an unrealistically high price. Therefore, we should use the mean squared error as our performance metrics. Thus, we take the mean of the sum of the squared errors/differences in order to estimate how close our predictions are to the actual values on average.

Other measurements like the classification metrics are not appropriate here because we are facing a regression and not a classification problem. The r2 score and explained variance score are not appropriate here either as they do not provide us with information on how accurate our predictions are.

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

  It is important to split the data into a training and a test set in order to get a true test of the model. If we weren't to split the data into a training and test set, our model would be trained on the whole data set. Therefore, we also chose the parameters on the whole data set. This could lead to overfitting as the algorithm tries to minimize the error on the training data. Because of the overfitting, we would think that we have found a good model as the training error would be low. However, our model might not perform well on unseen data as it is fit too much to the training data. Therefore, we need a test data set in order to truly evaluate our model's accuracy.

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

  Grid search allows us to test multiple parameters for one algorithm. By using Grid search we are able to automatically identify the best performing parameters without having to call the algorithm multiple times with the different parameters on our own.

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

  Cross validation is useful for determining the accuracy of our model's prediction. As mentioned earlier, if we weren't to split the data into a training and test set, our model would be trained on the whole data set. Therefore, we would choose the parameters on the whole data set. This could lead to a too optimistic evaluation of our model's accuracy as the model would perform worse on unseen data – even though the performance on the unseen data set is what we often are most interested in.

  Therefore, we should employ cross validation. Cross validation means that we are not using the entire data set when training our model. Instead, we set parts of our data aside (as a test set) and do not let the algorithm use those data when training the model. Once training is complete, we can use the data we originally set aside as our test set to test our model's performance on unseen data.

  In the simplest form of cross validation, the data gets separated into a training and a testing set. This approach is called the hould-out method. The model is trained on the training set only. After training, the model is used to make predictions on the test set. The individual errors on the test set are aggregated and are used to determine a model's accuracy. The upside of this approach is the relatively easy computation. However, the downside is that the evaluation of the test error can have high variance as the evaluation depends strongly on which data points were included in the test set (especially true for smaller data sets.)

  More sophisticated approaches to cross validation are k-fold cross validation and leave-one-out cross validation. Both implementations are similar to the hold-out method.
  However, instead of splitting the complete data set into one training set and one test set, k-fold cross validation splits the data multiple (k) times randomly so that the data set gets divided into k subsets. Each of the k subsets becomes the test sets once, while the remaining k – 1 subsets become the training sets. The model gets trained on the k -1 subsets and is then used for prediction on the test set. This process is repeated until every subset was used as the test set once. The prediction errors obtained on the test sets are then averaged to get the test error of the model. Because of the averaging of the test errors, k-fold cross validation reduces the variance of the resulting test error estimate; therefore, giving a better estimate for a

model's true performance on unseen data. With every data point being used for predictions exactly once, k-fold cross validation also helps to get a good estimate for our model's true performance – especially for small data sets. One of the disadvantages of this method though is that the algorithm has to run k-times to train the model. Therefore, it takes k-1 times longer to train the model.

The leave-out-one cross validation method takes k-fold cross validation to the extreme as it sets k to the number of data points in the data set. Therefore, the model gets trained on all data points except for the one data point that is then used for predictions. Similar to k-fold cross validation, the prediction errors are then averaged over all predictions, i.e. the predictions for every data point in the data set.

We might use cross validation with grid search because we are trying to find the best performing parameters with grid search. As the true performance should be determined based on performance on unseen data, we should use cross validation with grid search so that grid search identifies the parameters that perform best on the test set instead of on the whole data set. By using grid search together with cross-validation we reduce the variance in our test error estimate; therefore, we can be more confident in the parameters found by grid search.

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?

  The training error increases as training size increases, but only to a point after which the training errors levels off or only increases slightly. At first, the test error decreases strongly with increased training size. However, as the training size increases even more, the test error starts to decrease less strongly until it levels off.

- 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?

  At max depth of 1 the model suffers from high bias. Both training and test error are very high. The model with max depth of 10 suffers from high variance as the train error goes against 0, but the testing error remains high. This indicates that our model is overfitted to the training data and did not generalize well.

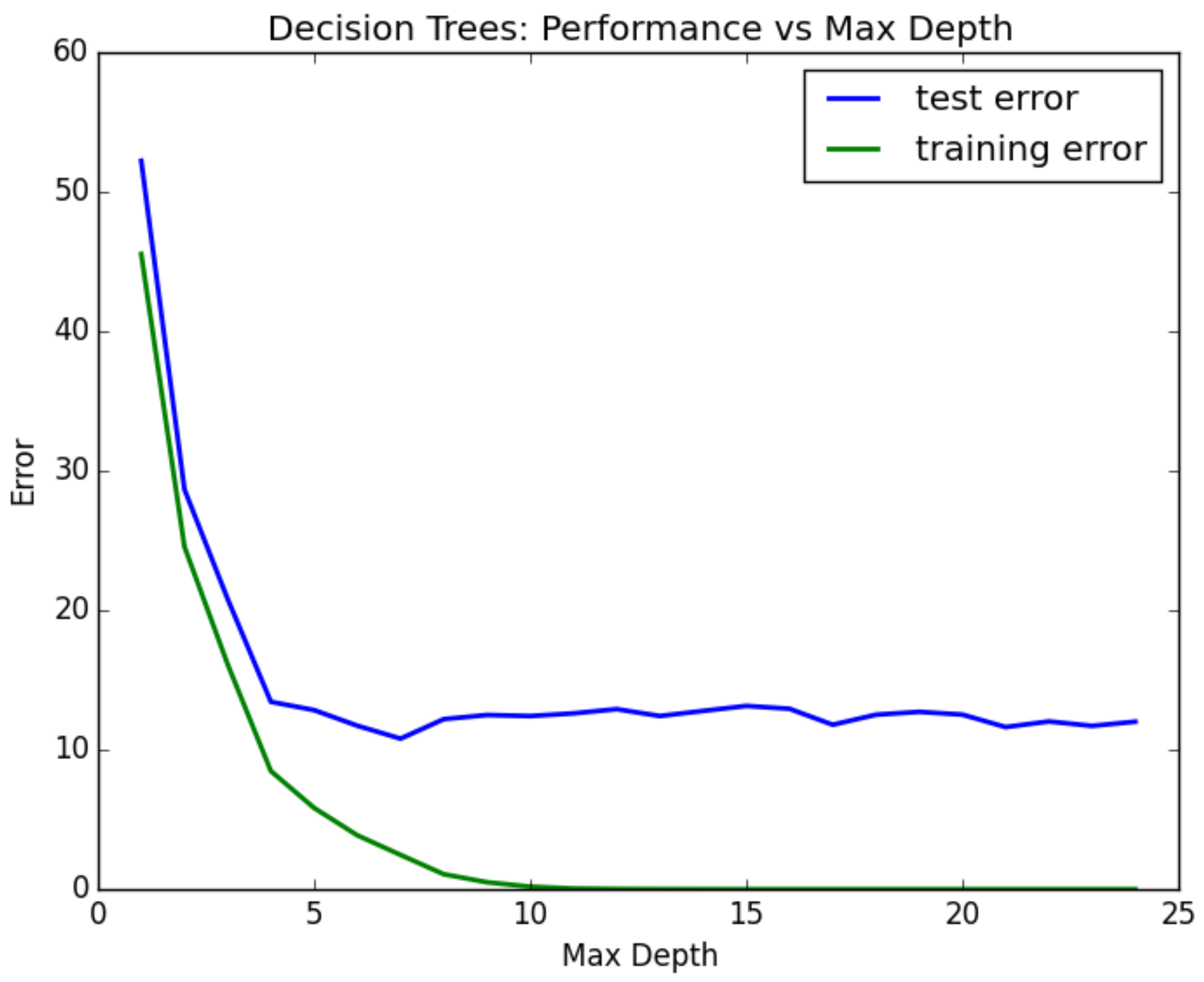
- Look at the model complexity graph. How do the training and test error relate to increasing model complexity? Based on this relationship, which model (max depth) best generalizes the dataset and why?

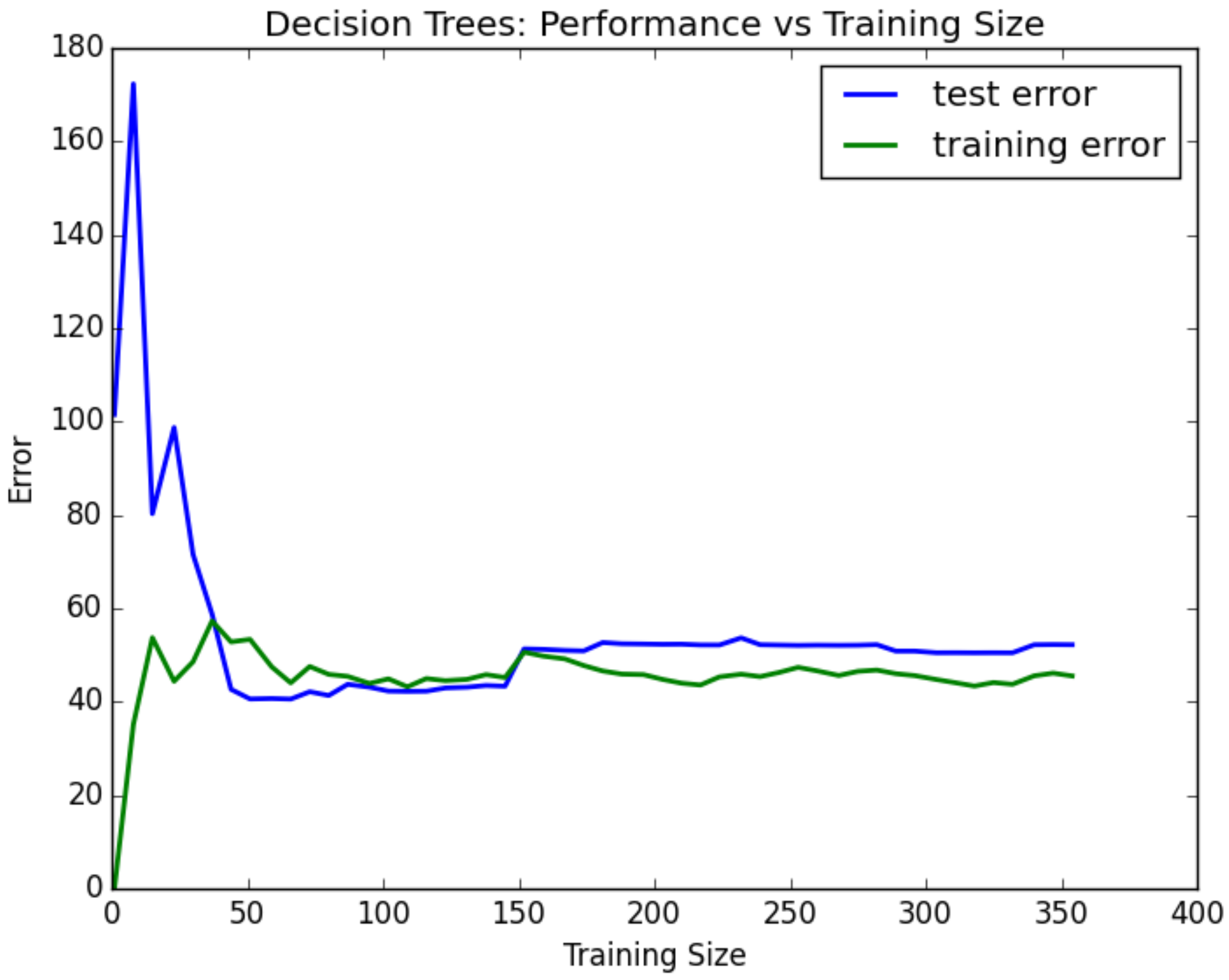
  Training and test error decrease as the model becomes more complex, i.e. until a max depth of 5. Afterwards (max depth of > 5), the test error remains around the same level while the training error continues to decrease. This behavior signals overfitting. Therefore, the model with max depth of 5 best generalizes the dataset – the test error remains around the same level afterwards while the gap between training and test error increases (signal for overfitting).

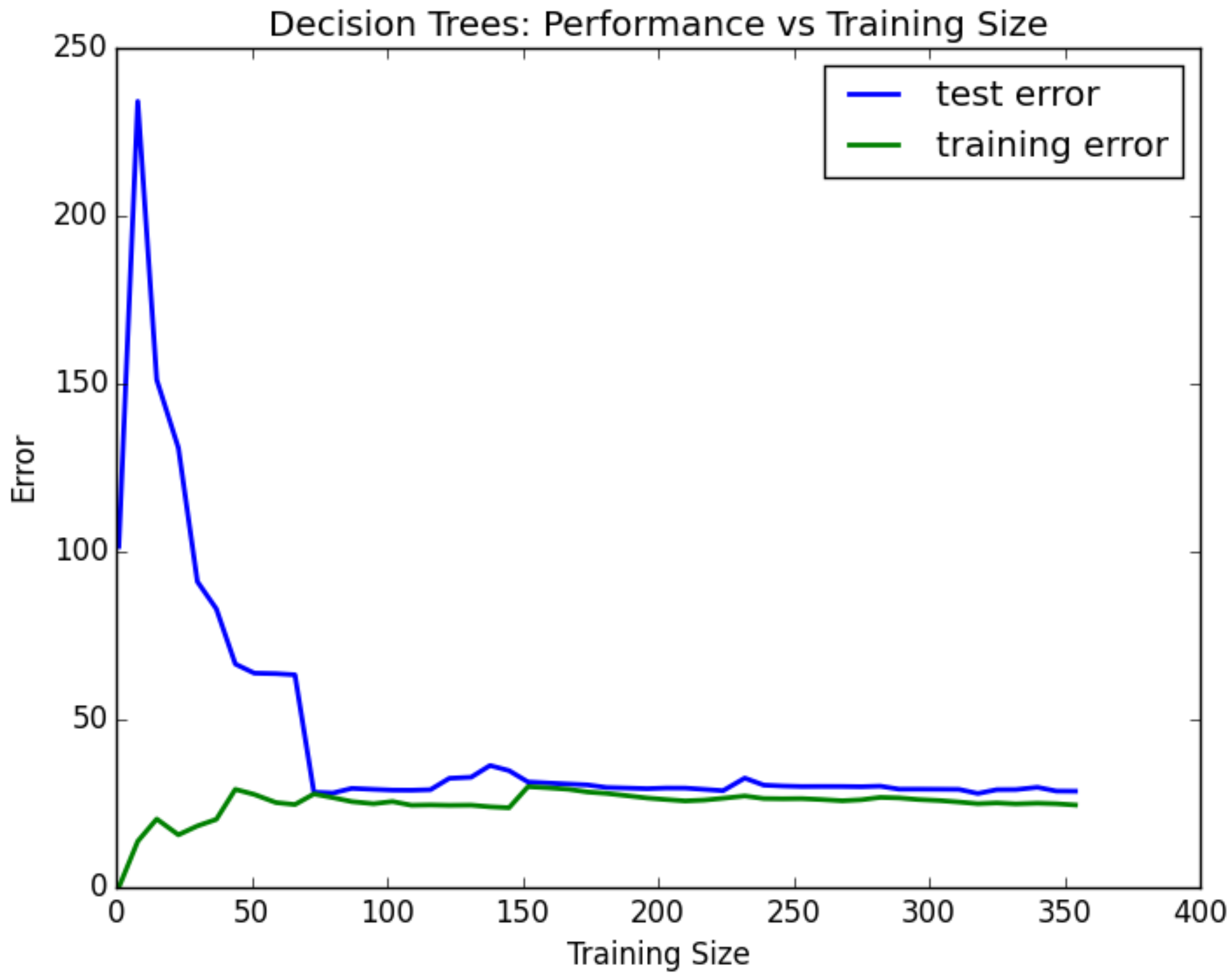
4) Model Prediction

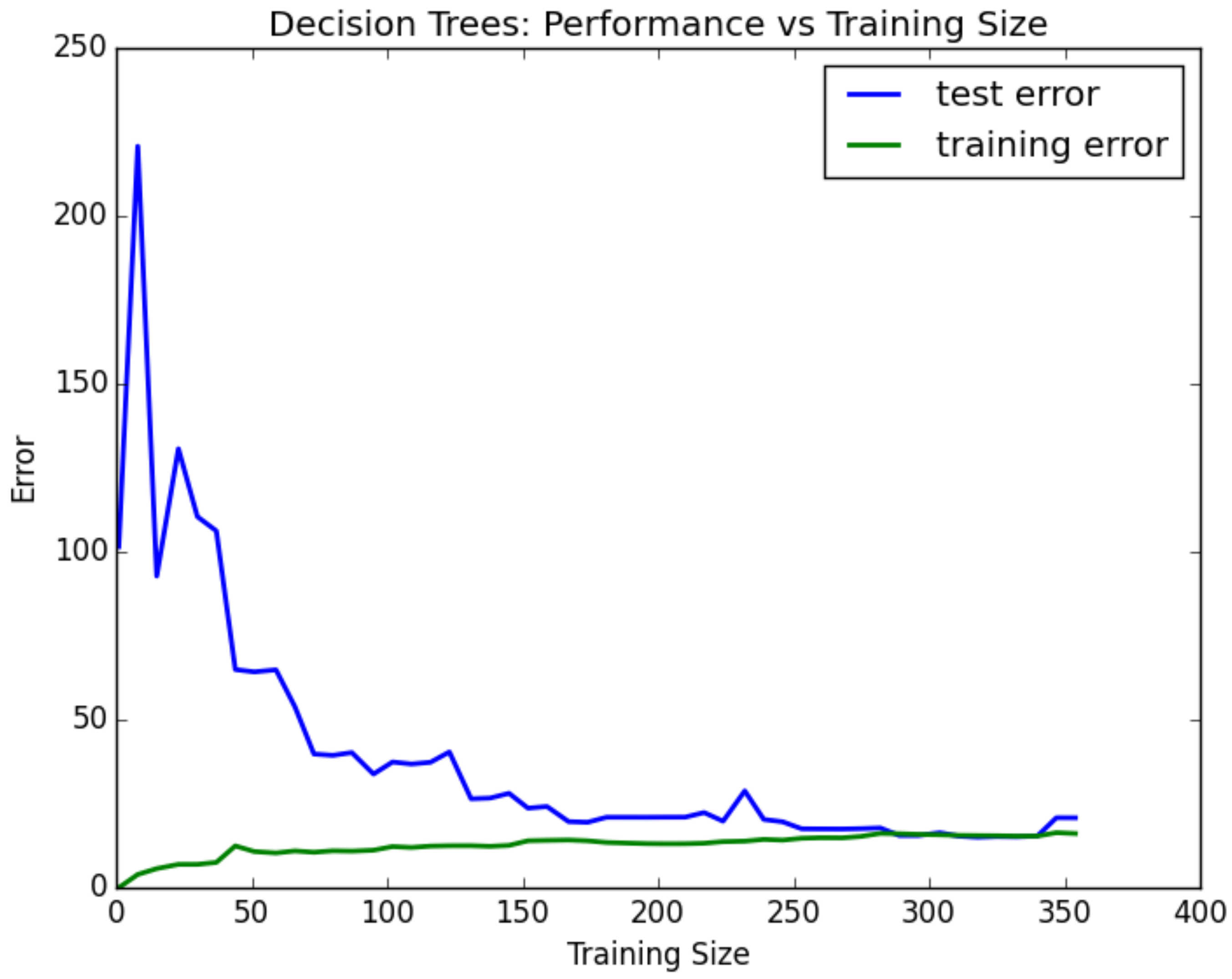
- Model makes predicted housing price with detailed model parameters (max depth) reported using grid search. Note due to the small randomization of the code it is recommended to run the program several times to identify the most common/reasonable price/model complexity.
- Compare prediction to earlier statistics and make a case if you think it is a valid model.

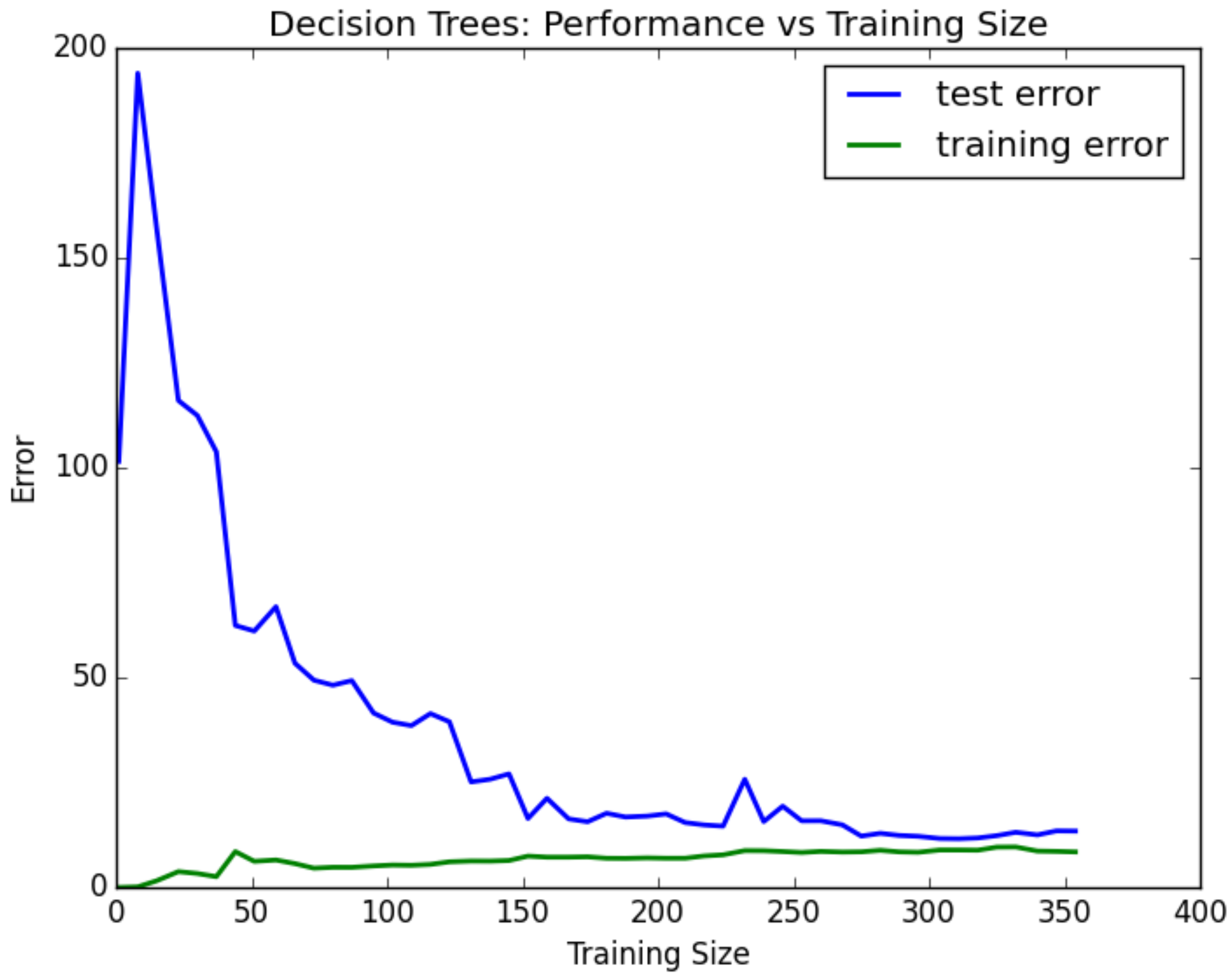
  During three runs, the model with a max depth of 5 predicted the following prices: 20.9677, 21.62974, and 20.766. These predictions lie within the range of the reported min and max values. They also lie within one standard deviation around the mean and median housing prices. Therefore, the predictions do not seem unlikely and we can assume the model to be a valid model.

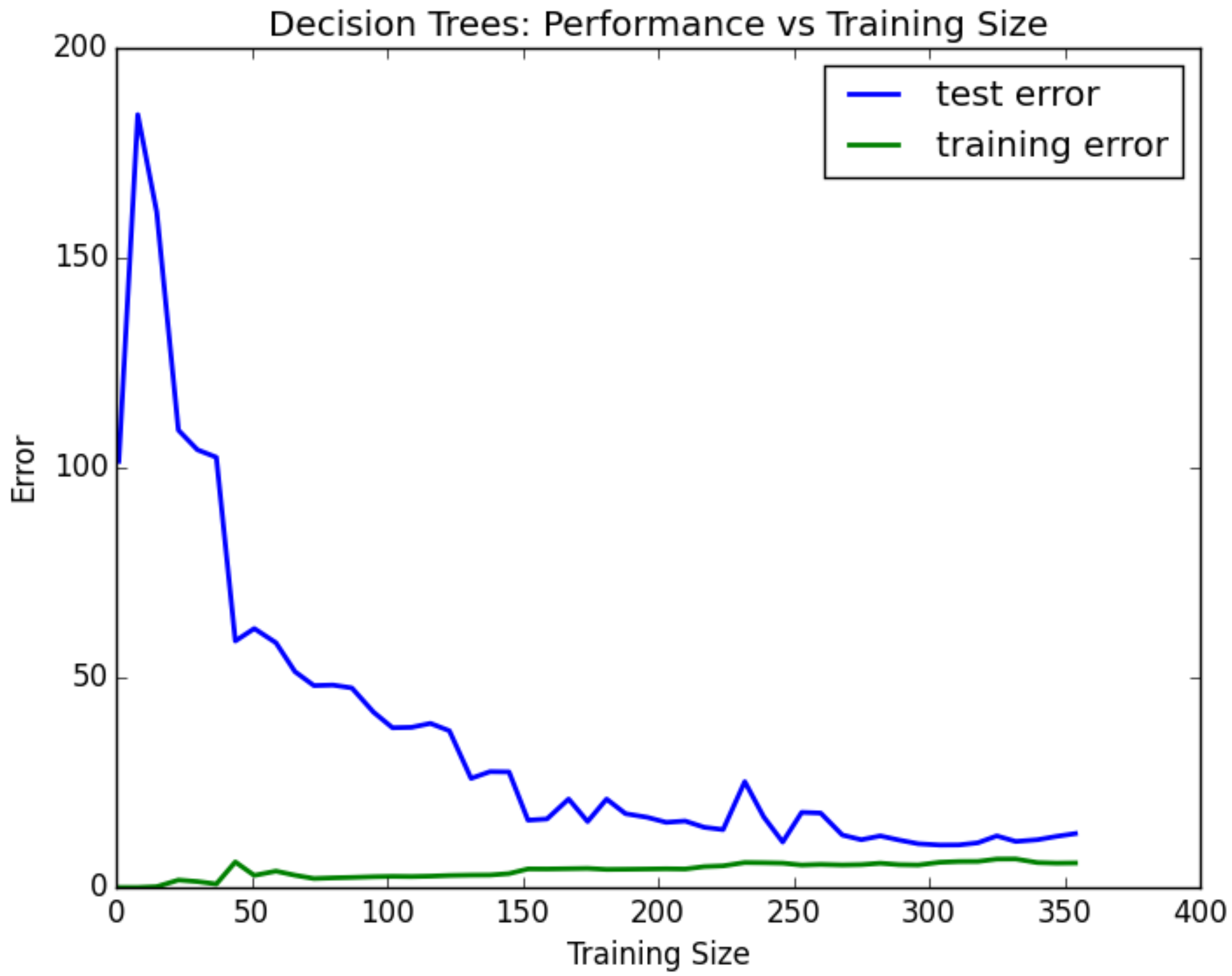

Decision Trees: Performance vs Max Depth
test error
training error
Error
Max Depth
0
10
20
30
40
50
60
0
5
10
15
20
25

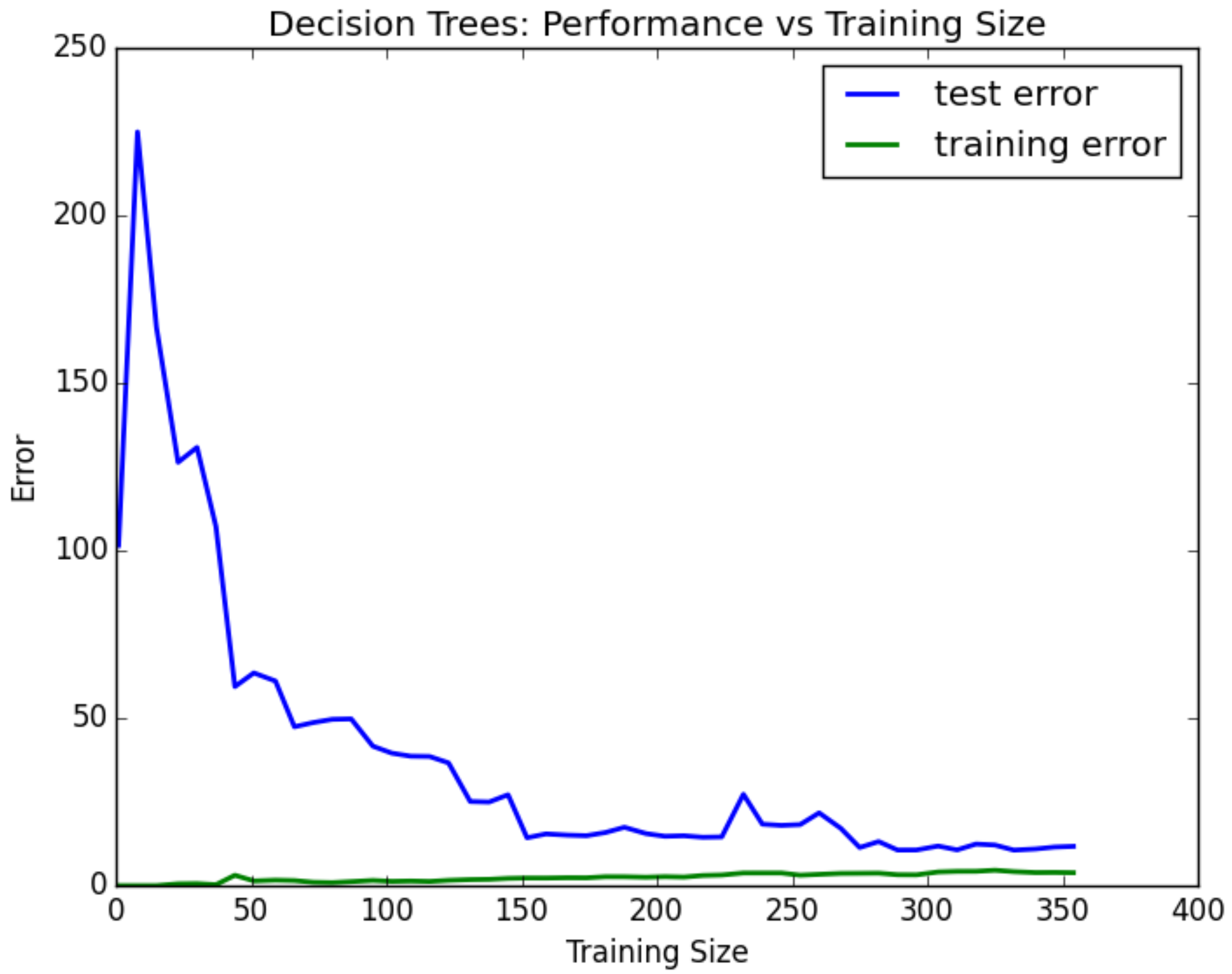

Decision Trees: Performance vs Training Size
test error
training error
Error
Training Size
0
20
40
60
80
100
120
140
160
180
0
50
100
150
200
250
300
350
400

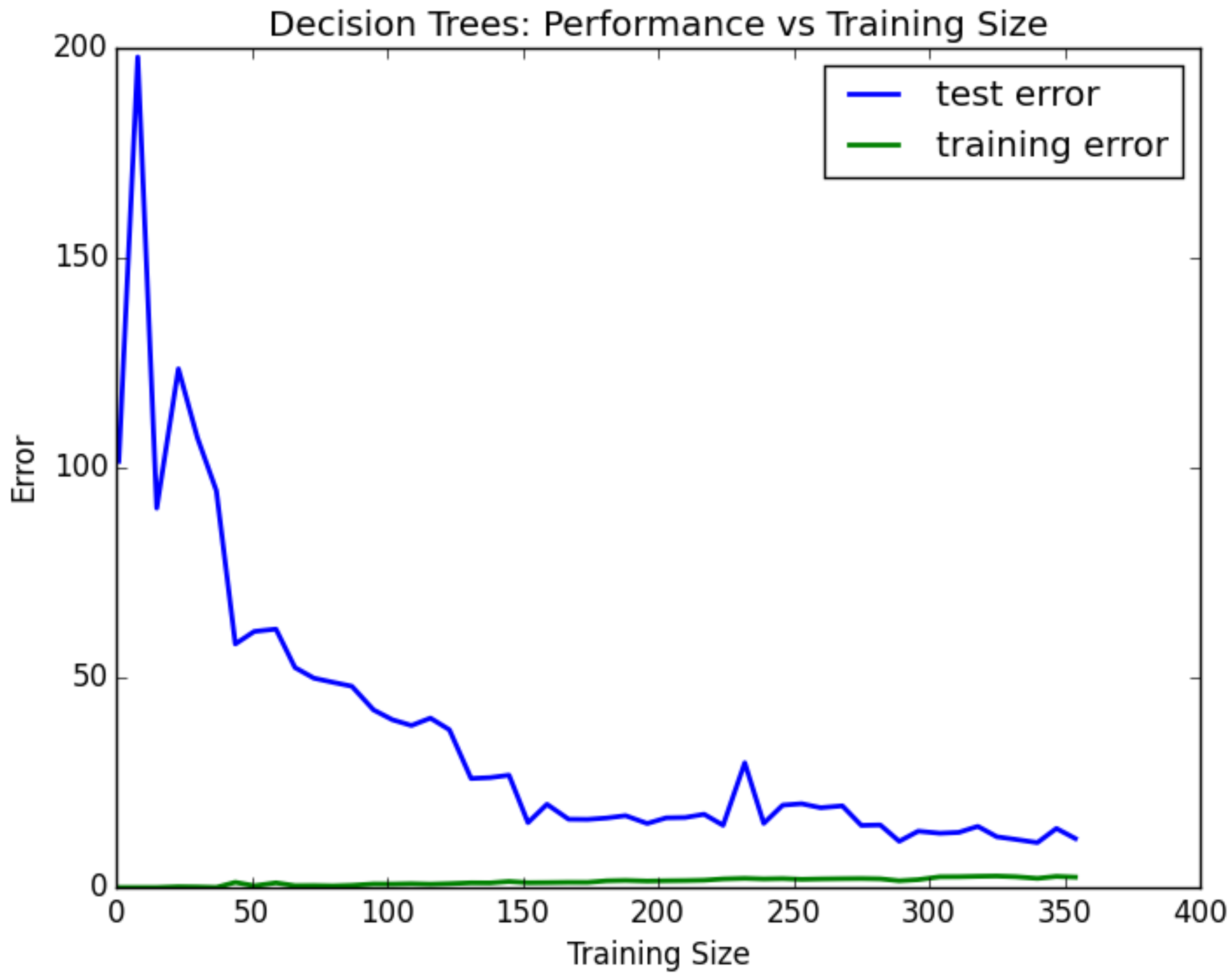

Decision Trees: Performance vs Training Size
Error
Training Size
test error
training error
0
50
100
150
200
250
300
350
400

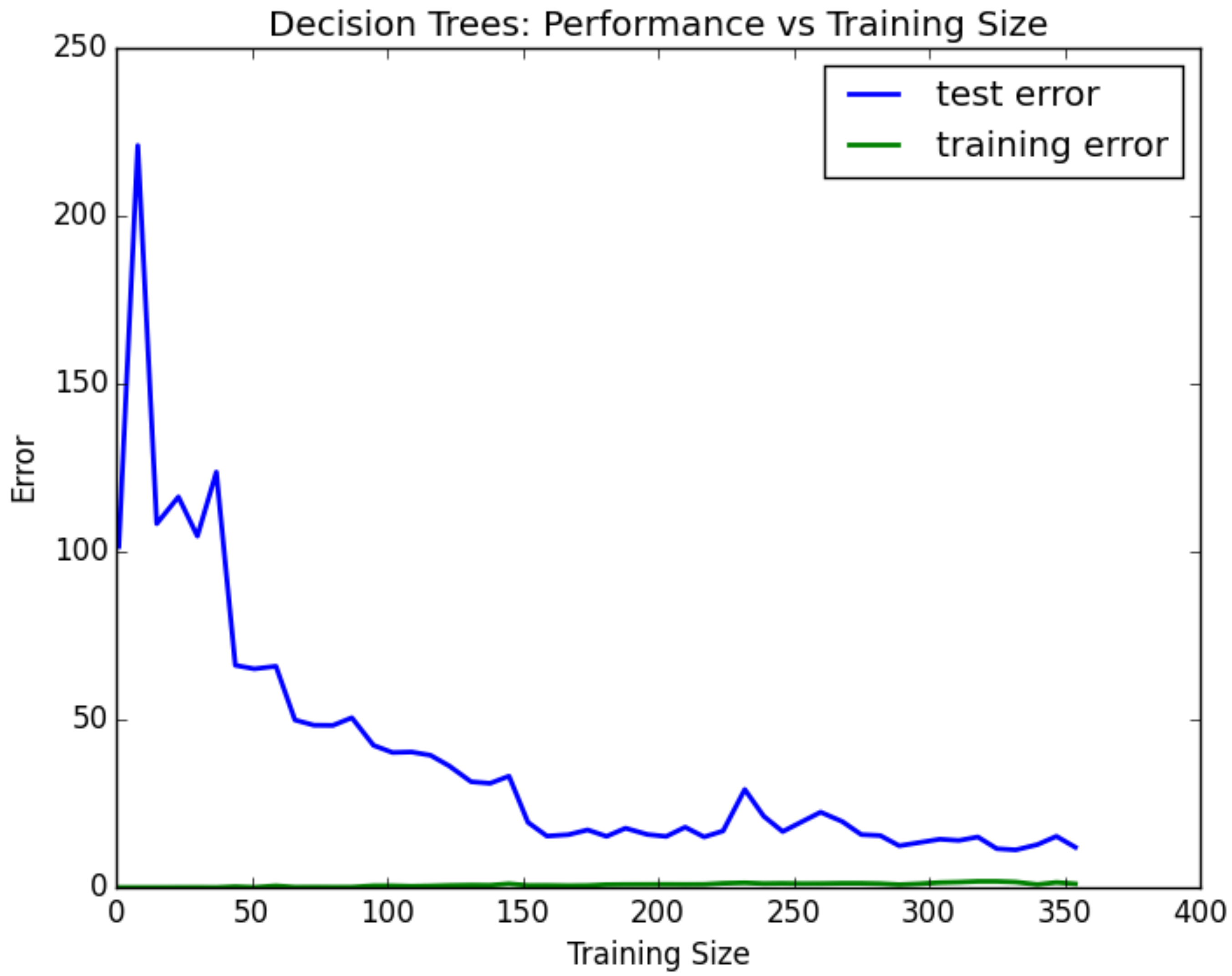

Decision Trees: Performance vs Training Size
test error
training error
Error
Training Size
0
50
100
150
200
250
0
50
100
150
200
250
300
350
400


Decision Trees: Performance vs Training Size
test error
training error
Error
Training Size
0
50
100
150
200
250
300
350
400

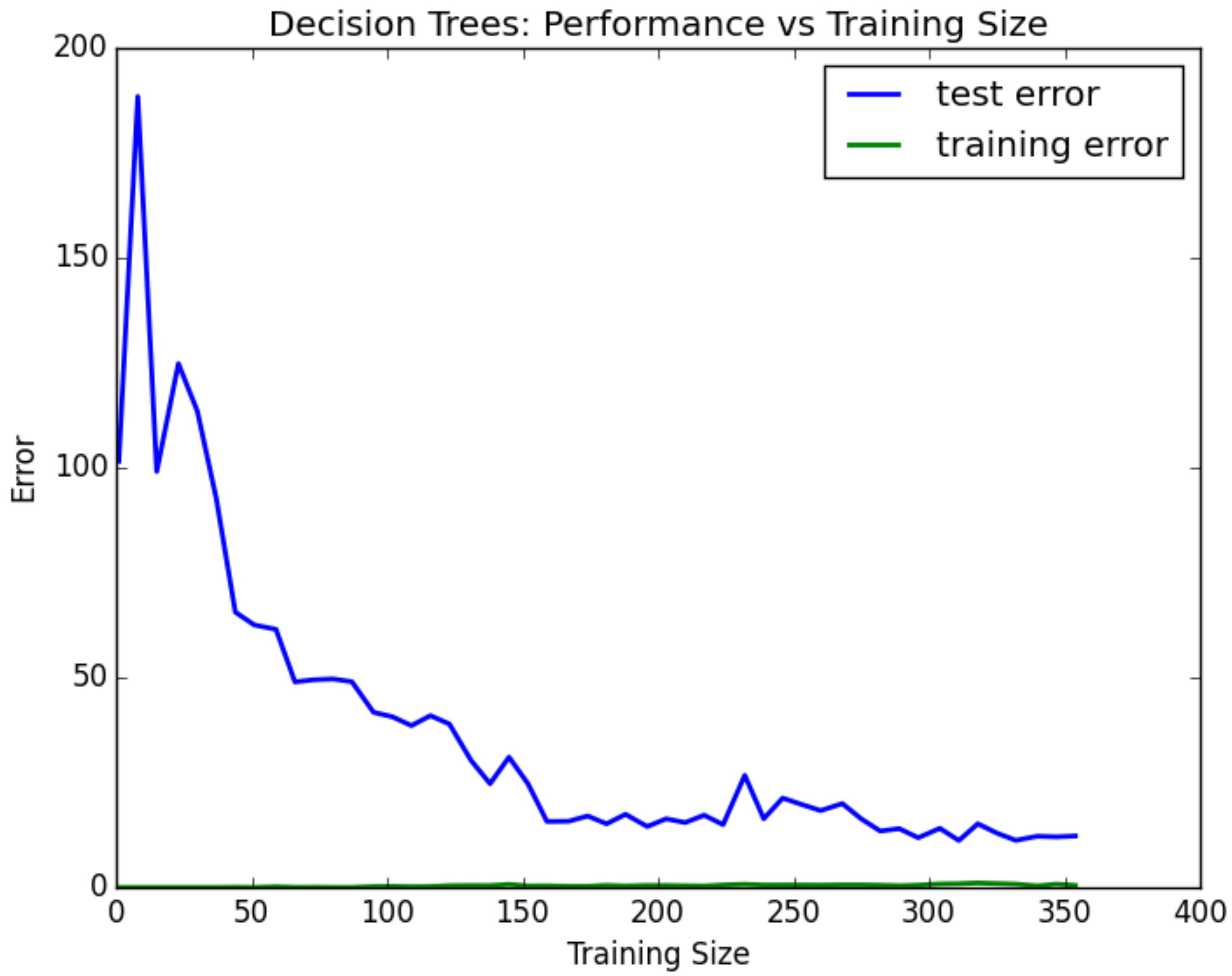

Decision Trees: Performance vs Training Size
Error
Training Size
test error
training error
0
50
100
150
200
250
300
350
400

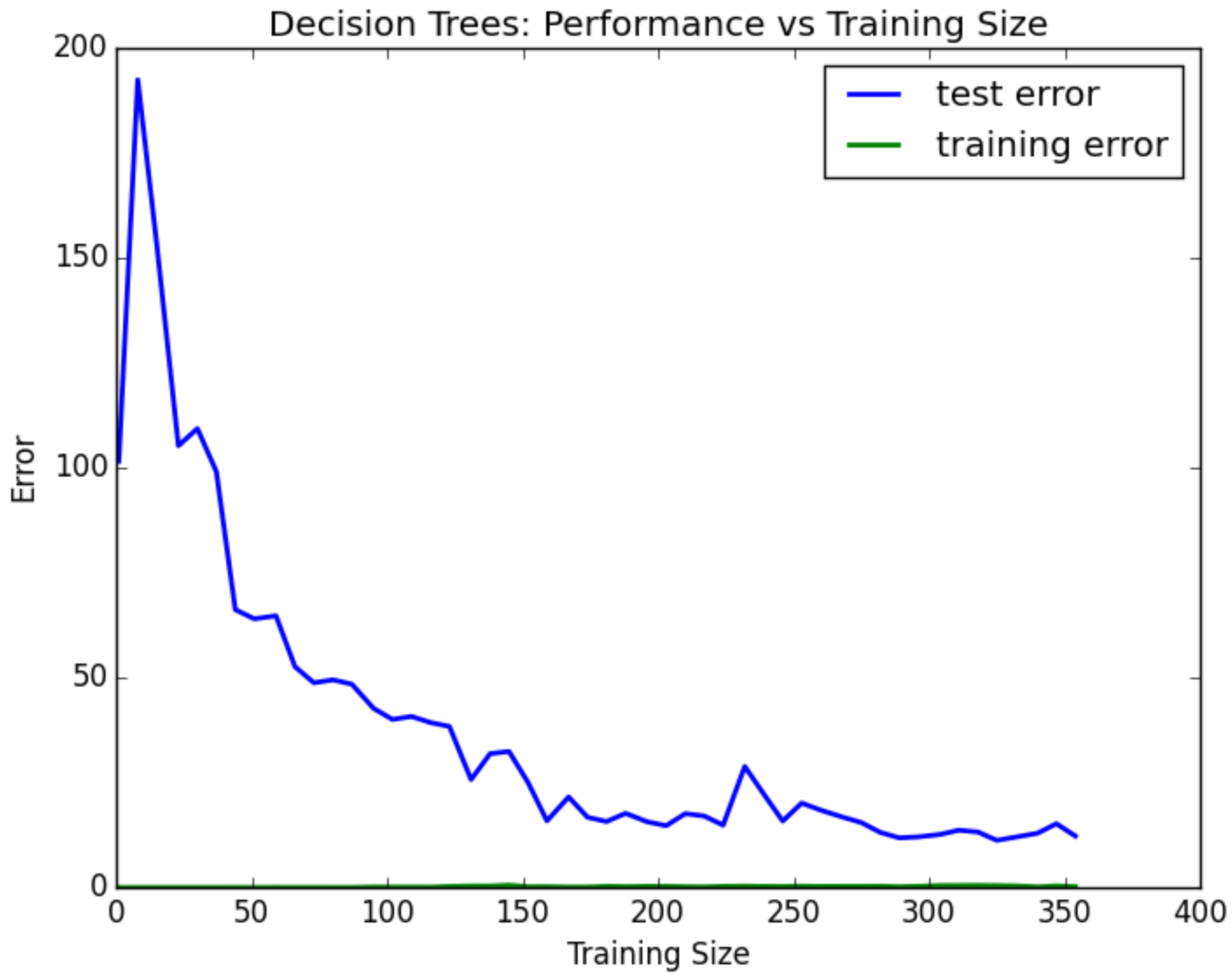

Decision Trees: Performance vs Training Size
Error
Training Size
test error
training error
0
50
100
150
200
250
300
350
400


Decision Trees: Performance vs Training Size
test error
training error
Error
Training Size
0
50
100
150
200
0
50
100
150
200
250
300
350
400



Decision Trees: Performance vs Training Size
test error
training error
Error
Training Size
0
50
100
150
200
250
0
50
100
150
200
250
300
350
400


Decision Trees: Performance vs Training Size
test error
training error
Error
Training Size
0
50
100
150
200
250
300
350
400


Decision Trees: Performance vs Training Size
test error
training error
Error
Training Size
0
50
100
150
200
250
300
350
400
0
50
100
150
200