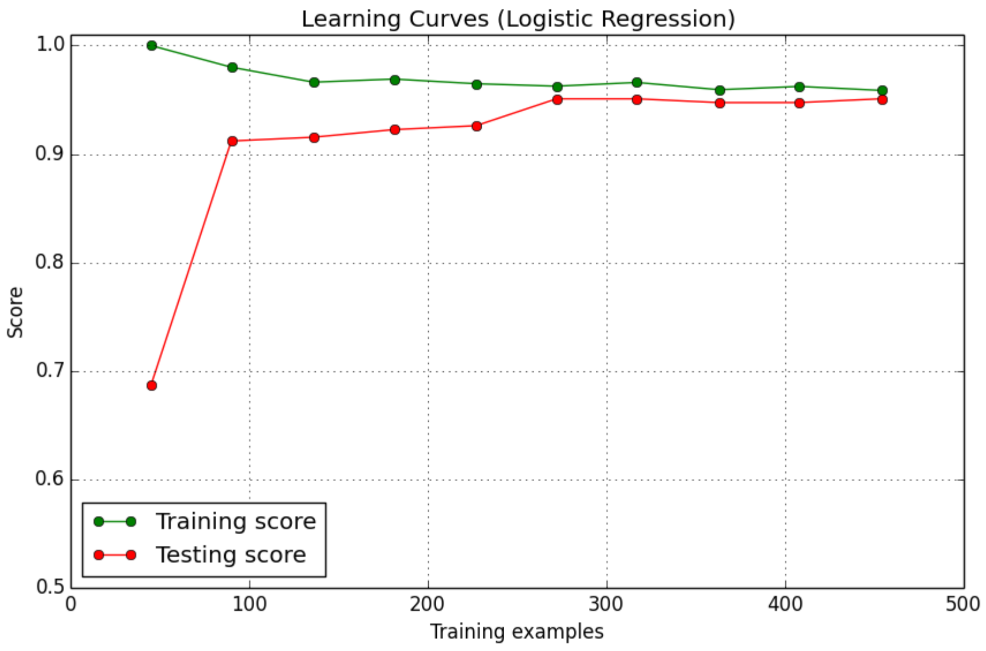
Report

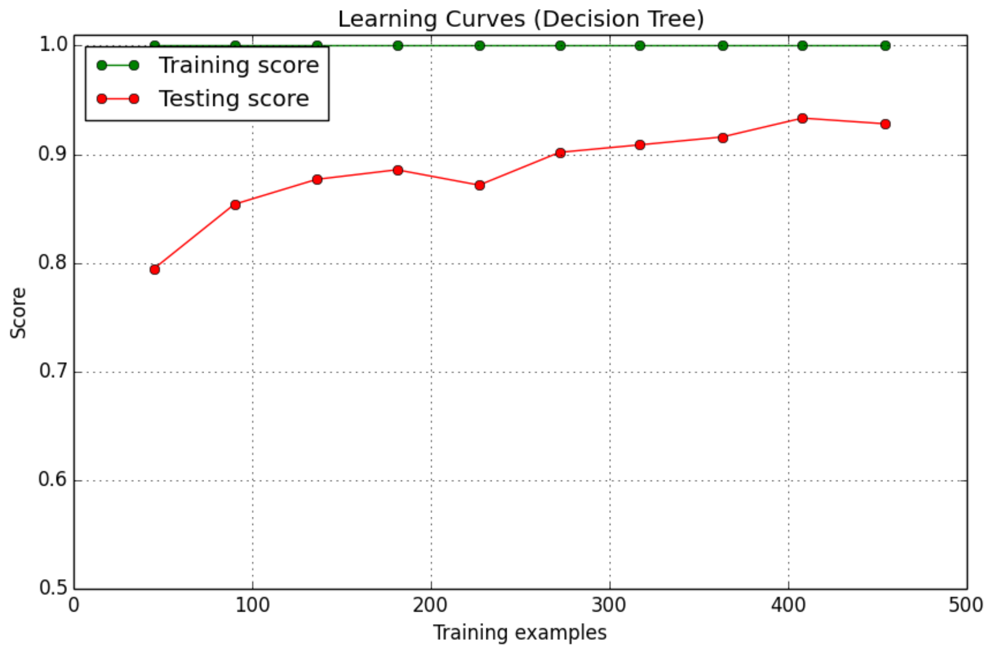
I use three approaches(logistic regression, decision trees, k-nearest neighbors) to classify the data.

Using two ways to draw the learning curve, one way is finding the optimal parameters for each classifies first, then fit the model of these optimal parameters, then for different sample size, draw the learning curve. The other way is TA mentioned on piazza, for each sample size, find the optimal parameters for this sample, then test the rest of data on this model. Then getting one point on the learning curve.

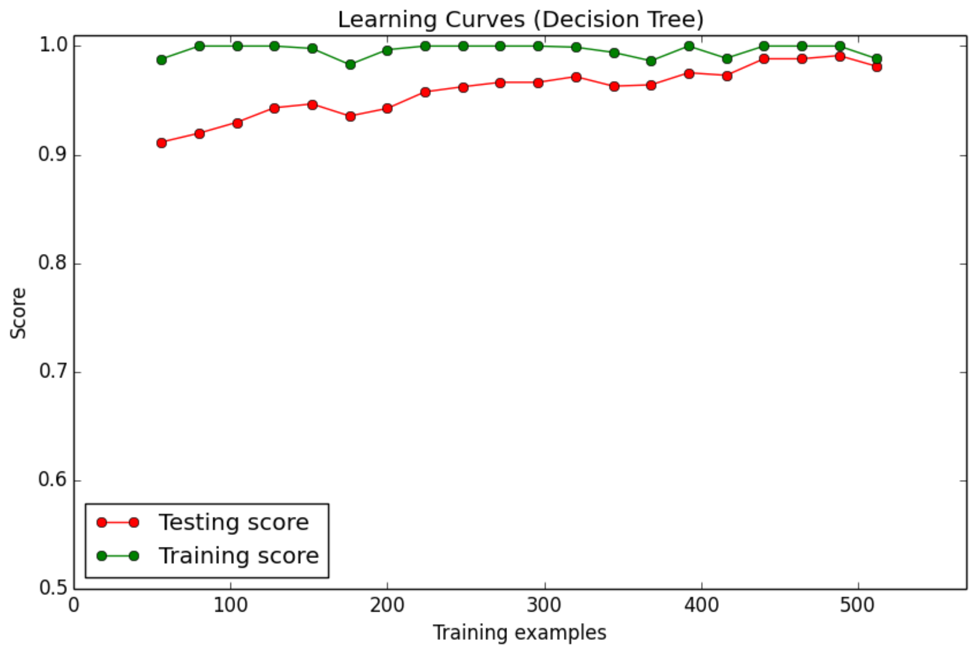
For logistic regression, there aren't parameters need us to fit in. So the only approach is showed in HW_LX7.ipynb. And the result is showing in LogReg.phg. The learning curve for this classifier is pretty nice. And I also tried using penalty L1 and L2 to solve the high variance problem in sklearn's library.(LogPenalty.ipynb) But since this logistic model don't have problem of high variance, using or not using penalty doesn't really make a difference.



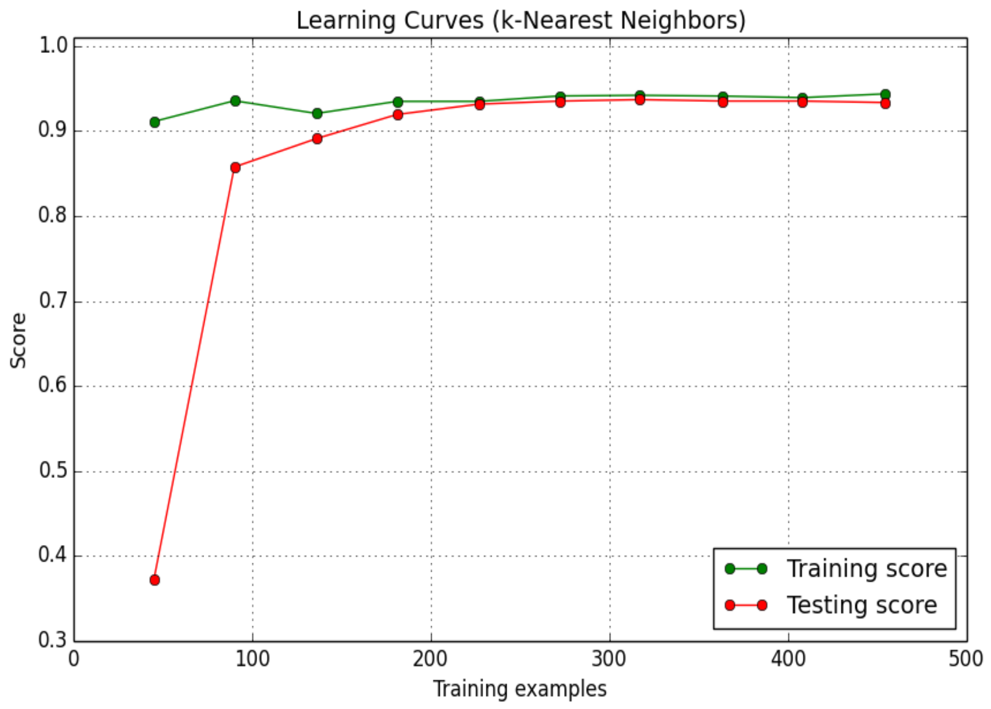
For decision tree, there are two approaches to find the optimal parameter. Finding optimal parameters over the whole dataset is one approach, after running the algorithm several times, the optimal parameters turns out to be criterion = entropy, and max_depth = 23. Using this parameters, the learning curve is showing as follows. But the problem is, due to the randomization of the algorithm, the best parameters are always different. I just pick one with most showing time. But the variance for the optimal parameter is high, so every time I may get a total different answer as the one I'm showing right now.(XL_HW7.ipynb)



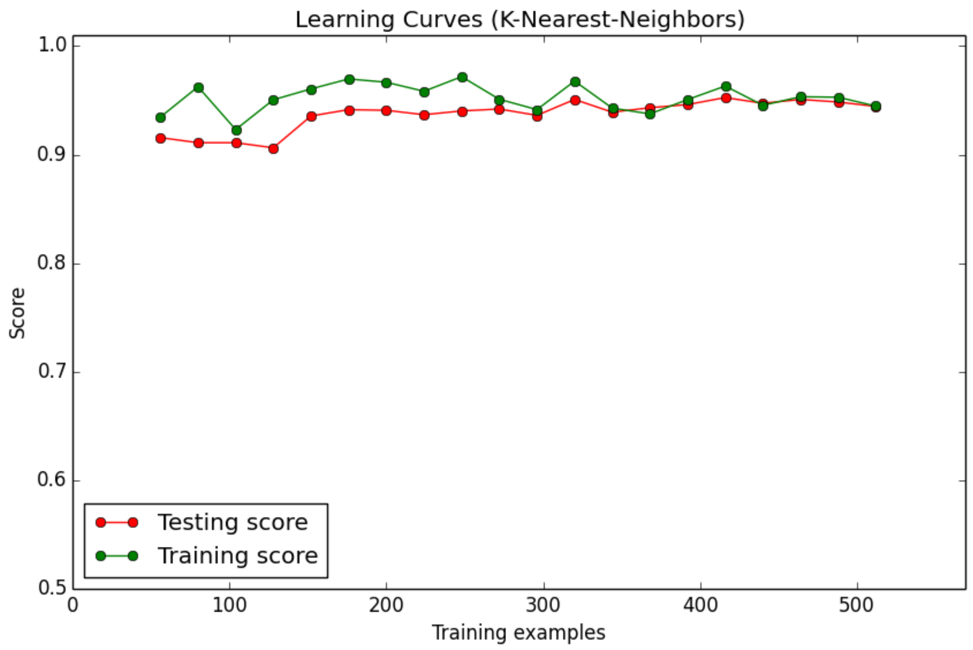
If using Liwen's approach to draw the learning curve, for each sample size, choose the best parameter for each sample size, then testing the model on the whole data set, the learning curve is showing as follows, from which we can see that the score on test data is actually much higher than the former one, because for each different sample size, we are choosing the best parameters.(TALearningCurve.ipynb)


Learning Curves (Decision Tree)
Score
Training examples
1.0
0.9
0.8
0.7
0.6
0.5
0
100
200
300
400
500
Testing score
Training score

For k-nearest neighbor algorithm, I'm using two approaches too. The one finding optimal parameters for the whole data is Neighbors = 13, and p = 1. The learning curve is showing as following. As we can see from the learning curve, the parameter work so well that even the sample is not that large, the testing data's score is as high as the training data. (XL_HW7.ipynb)



Using Liwen's method, I get the learning curve showing as following, the same as Decision Tree, the average testing score is higher than last approach. But the problem is either for training data or testing data, the score don't get to 1. Because there are some under fitting happens. Maybe we should try to get more polynomial features in this circumstance. (TALearningCurve.ipynb)



In conclusion, compared these three methods, decision tree is the best one. By choosing the optimal parameters for the model, although there are some high-variance problem, try to get more sample data may fix this problem. Or try smaller set of features.

So I try use the forward searching method try to select the best feature for the problem(BestFeatureSelection.ipynb). By fix the parameters of the model, I try to get the highest score with this number of features. And as a result, by selecting the former 22 features other than the whole 32 features can reduce the high variance problem.