**Wine Quality Dataset**

I picked a wine dataset to understand what determines the quality of wine. I want to know what ingredients and what chemical properties to pay attention to when buying a bottle of good wine. Of course, one can look it up online and approximately know what those are. However, it would be nice to have a model to plug those properties into and get an estimate on the wine quality.

**Banknote Authentication Dataset**

I find this dataset interesting because of a pure interest as to how a forged banknote can be told from a genuine one. Can one distinguish between those just by looking at them? Or is it only through a thorough analysis in the lab that one can tell which one is which.

**Decision Tree - *Wine***

The chosen dataset has 4,898 entries which could be a problem if the target classes are unbalanced. The following are the features:

**fixed\_acidity, volatile\_acidity, citric\_acid, residual\_sugar, chlorides, free\_sulfur\_dioxide, total\_sulfur\_dioxide, density, pH, sulphates, alcohol, Class**

The target class to be predicted includes 7 discrete values ranging from 1 to 7. Let us check the distribution of those classes in the whole dataset:

df["Class"].value\_counts()gives us the following output:

**4 2198**

**3 1457**

**5 880**

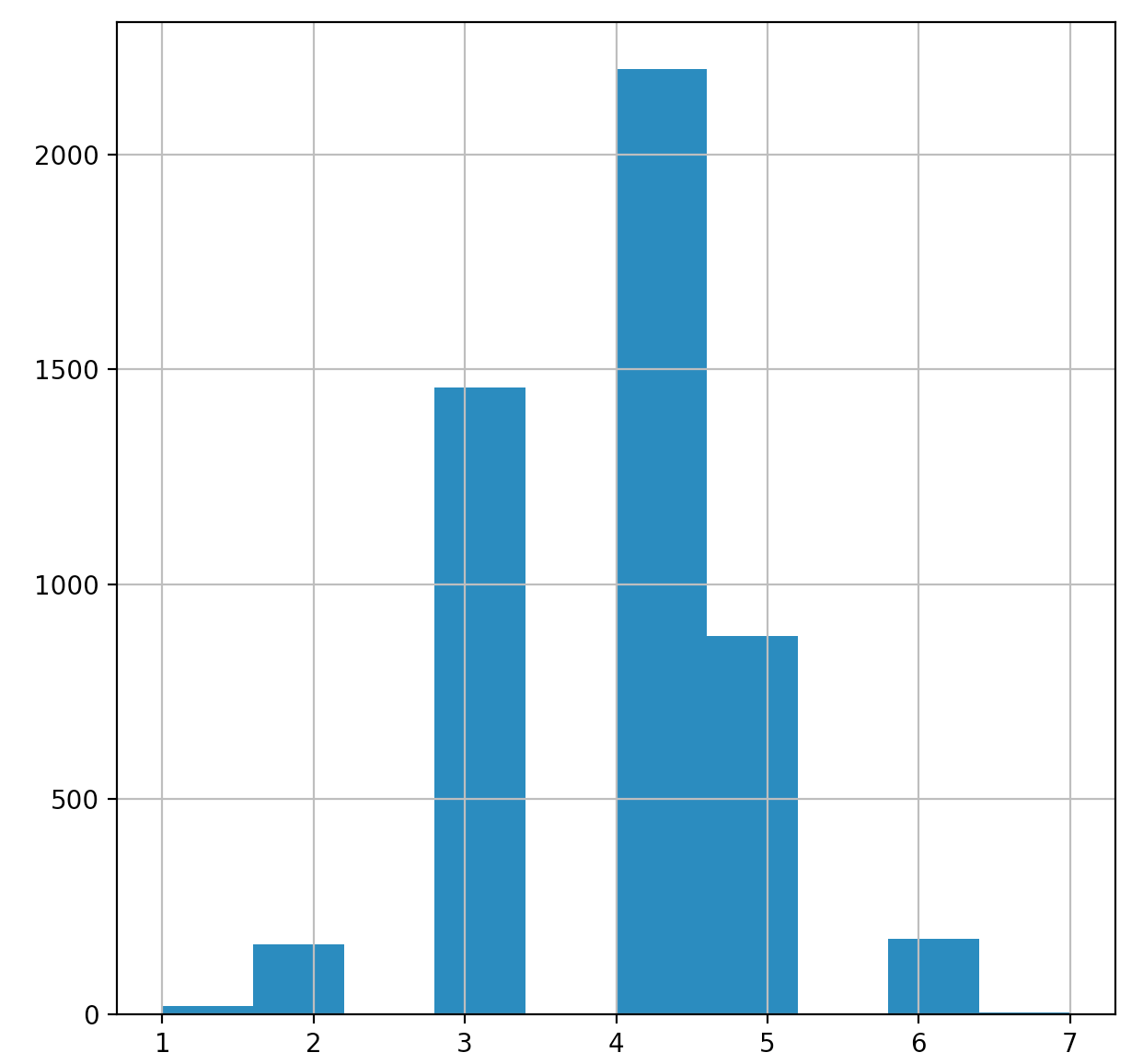
**6 175**

**2 163**

**1 20**

**7 5**

And this is what the histogram looks like:



Since the dataset is highly unbalanced, I am going to use StratifiedShuffleSplit from the sklearn library and allocate 20% of the data for a test set.

I will be using the DecisionTreeClassifier(), as it can perform multi-class classification. Let us run a 10-fold cross-validation and see the score:

**Mean: 0.594696952651**

**Standard Deviation: 0.0141753636395**

It seems that the mean class is 6, which does not look quite right.

After fitting the train data to the classifier and then predicting the output, we see the accuracy rate of about **60%**.

We can also calculate per-class recall:

**Class 4 : 0.627272727273**

**Class 3 : 0.601374570447**

**Class 5 : 0.528409090909**

**Class 6 : 0.6**

**Class 2 : 0.242424242424**

**Class 1 : 0.0**

**Class 7 : 0.0**

These recall values (along with precision) can also be retrieved by using skilern’s confusion matrix:

**precision recall f1-score support**

**1 0.00 0.00 0.00 4**

**2 0.27 0.27 0.27 33**

**3 0.61 0.62 0.61 291**

**4 0.66 0.63 0.65 440**

**5 0.54 0.56 0.55 176**

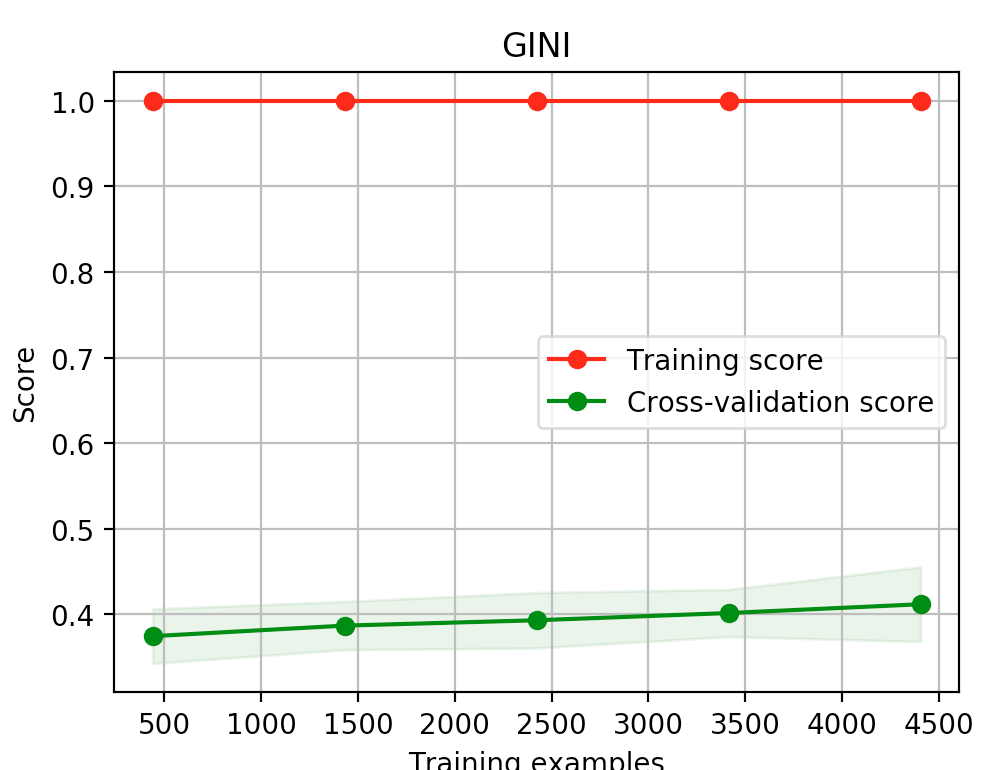
**6 0.40 0.57 0.47 35**

**7 0.00 0.00 0.00 1**

**avg / total 0.60 0.60 0.60 980**

The classifier completely fails to predict classes 1 and 7, which are very few in the dataset.

Let us now train the model using different train sizes and plot a graph. Her I use the default **gini** criterion to measure the quality of split:



From the above we see that as the training size grows the prediction error decreases.

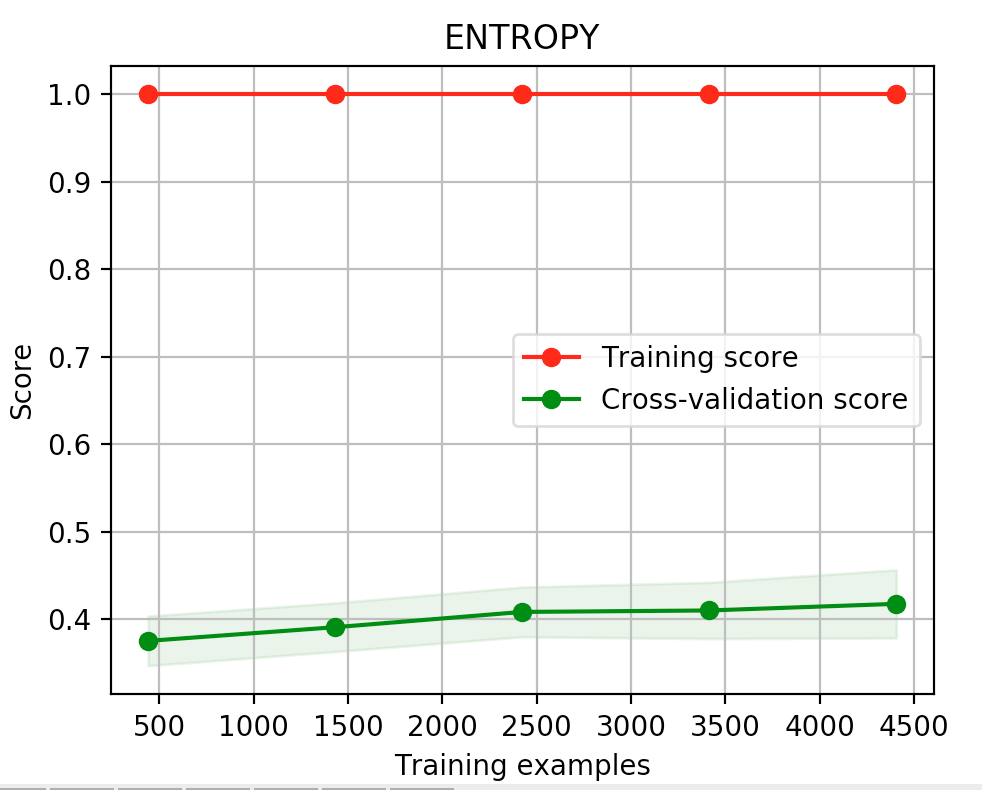
With this highly unbalanced dataset we can transform the labels to binary class such that wine classes less than or equal to 4 are considered bad, and those above 4 are considered good. After training and testing the classifier on just two classes, we have the following scores:

**Cross-validation mean score: 0.820301814719**

**Cross-validation standard deviation score: 0.0218074162124**

**Accuracy: 0.823469387755**

And this is where I use the entropy criterion:

****

**Decision Tree – *Banknote Authentication***

The dataset consists of 1,372 entries and has the following features: **variance of Wavelet Transformed image, skewness of Wavelet Transformed image, curtosis of Wavelet Transformed image, entropy of image, Class.**

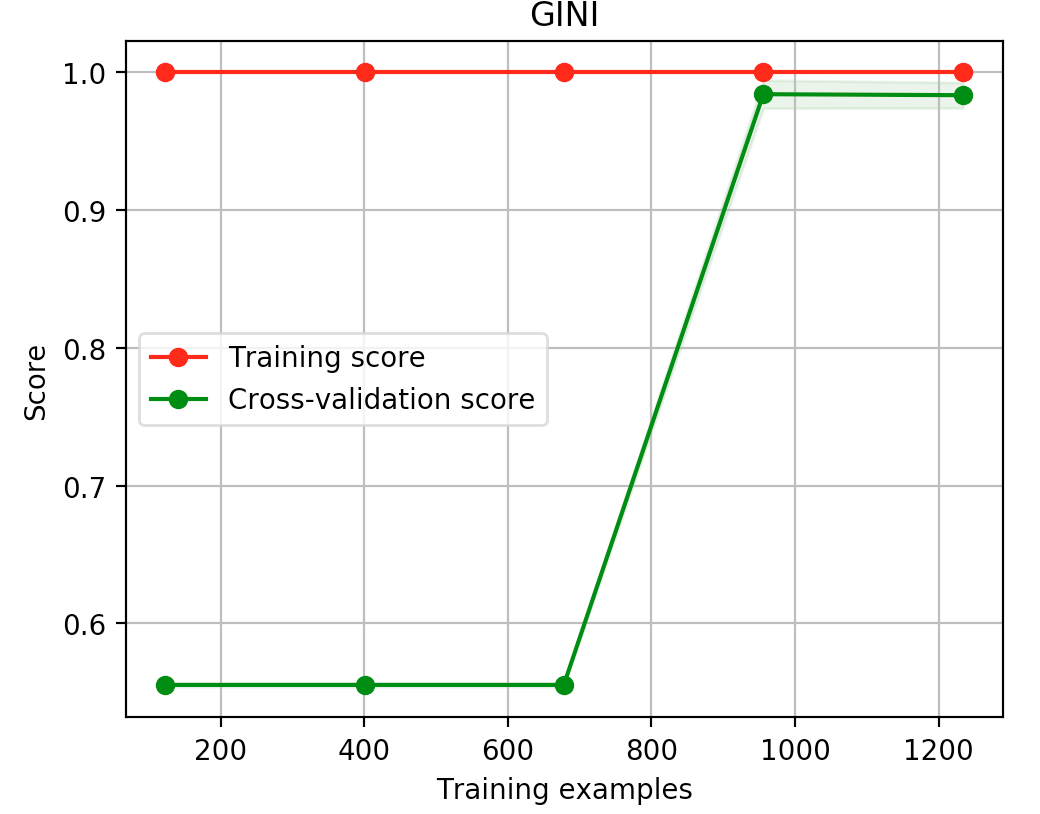
We need to predict the Class attribute, which it binary: {0,1}, i.e. whether a banknote is forged or genuine. Let us check the ratio of those classes:

**0 762**

**1 610**

The dataset appears well balanced. Let us know train the model on various train set sizes and check its performance on the test set.

DecisionTreeClassifier with the gini criterion:



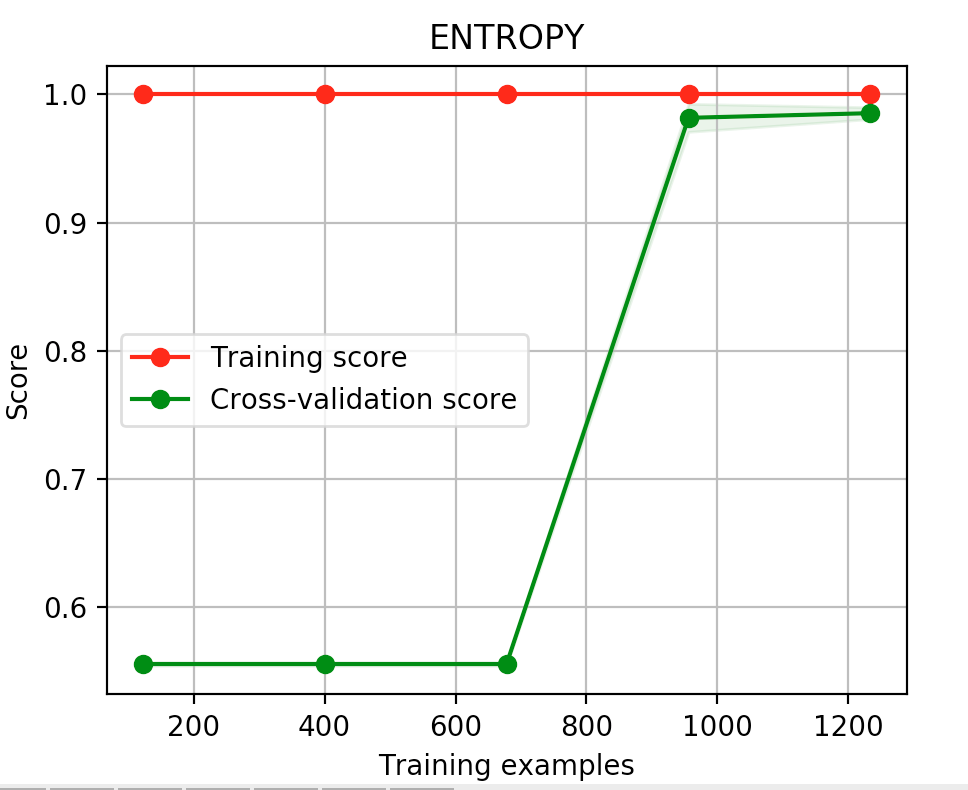
The 10-fold cross-validation of our classifier on the training set of size 80% is showing the following mean and std scores:

**Mean score: 0.989049053223**

**Standard deviation score: 0.00983019619596**

The mean accuracy score on the training data looks great. And this is the performance of the model on the test data: **0.992727272727**

DecisionTreeClassifier with the entropy criterion:



**Neural Networks - *Wine***

I am going to use sklearn’s MLPClassifier as a neural network classifier. It implements a multi-layer perceptron and trains using backpropagarion. Since MLP is sensitive to feature scaling, I will make use of StandardScaler to scale both the training and test sets. If we run df.describe().transpose(), we will see that two features especially stand out:

**mean std min max**

**free\_sulfur\_dioxide 35.455207 17.118267 2.00000 289.00000**

**total\_sulfur\_dioxide 138.384380 42.451829 9.00000 440.00000**

As before, for this dataset I will be using stratified split.

The number of hidden layers will be equal to 3 with 13 hidden units in each. Let us train the model and then predict the output using the test set.

Let us take a look at the 10-fold cross-validation report:

**10-fold cross-validation mean score: 0.549326030041**

**10-fold cross-validation standard deviation score: 0.0196155531156**

This is what the classification report is showing:

**precision recall f1-score support**

**1 0.00 0.00 0.00 4**

**2 0.50 0.21 0.30 33**

**3 0.55 0.53 0.54 291**

**4 0.54 0.60 0.57 440**

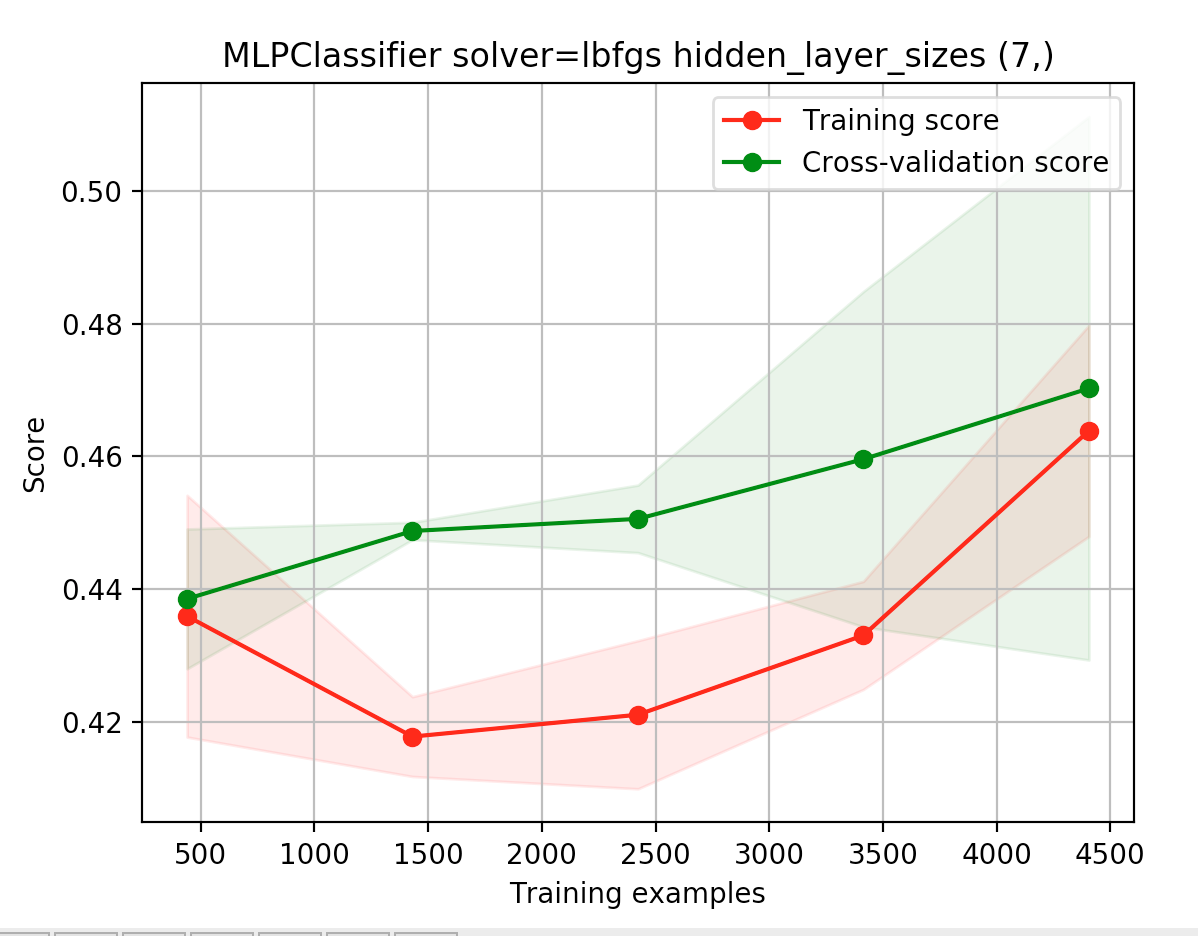
**5 0.45 0.49 0.47 176**

**6 0.60 0.09 0.15 35**

**7 0.00 0.00 0.00 1**

**avg / total 0.53 0.53 0.52 980**

Here is how the error rate changes depending on the training size:



The MLPClassifier did not do better than the DecisionTreeClassifier. As it was pointed out the data is highly unbalanced. For example, classes 1 and 7 are so few in the dataset that none of the two classifiers can predict those.

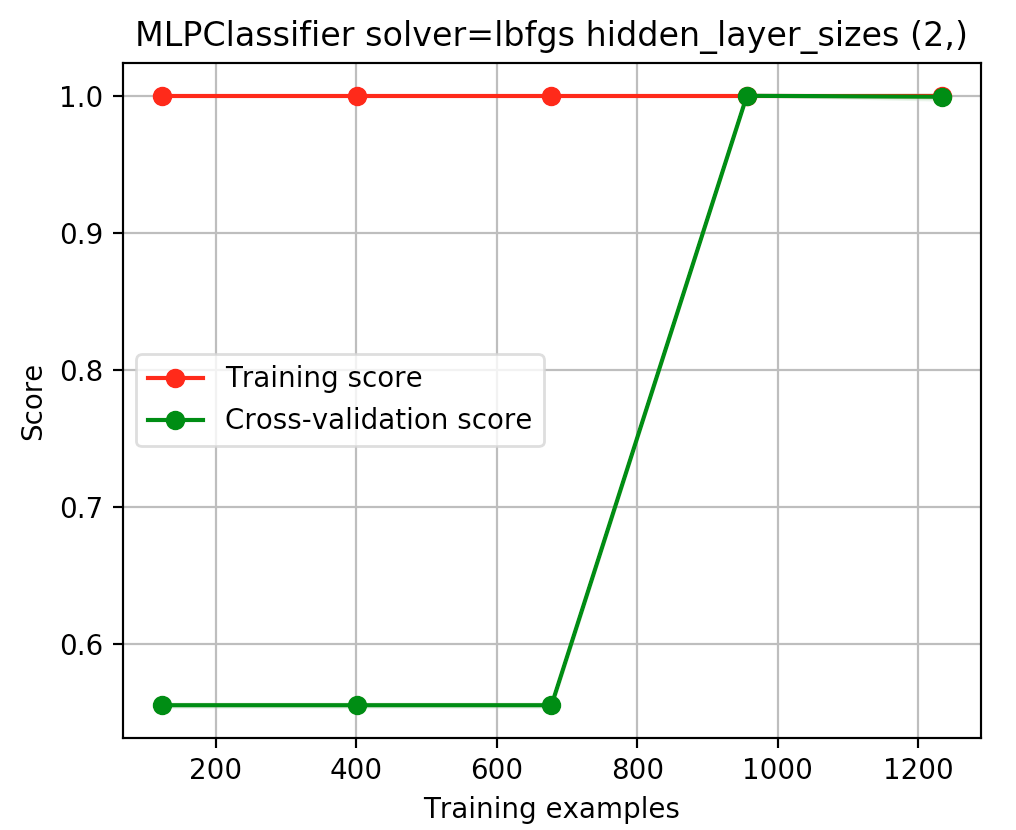
**Neural Networks – *Banknote Authentication***

Cross validation:

10-fold cross-validation mean score: 0.998173477898

10-fold cross-validation standard deviation score: 0.00365309180755

Training size vs error rate:



Prediction on the test data:

**precision recall f1-score support**

**0 1.00 1.00 1.00 153**

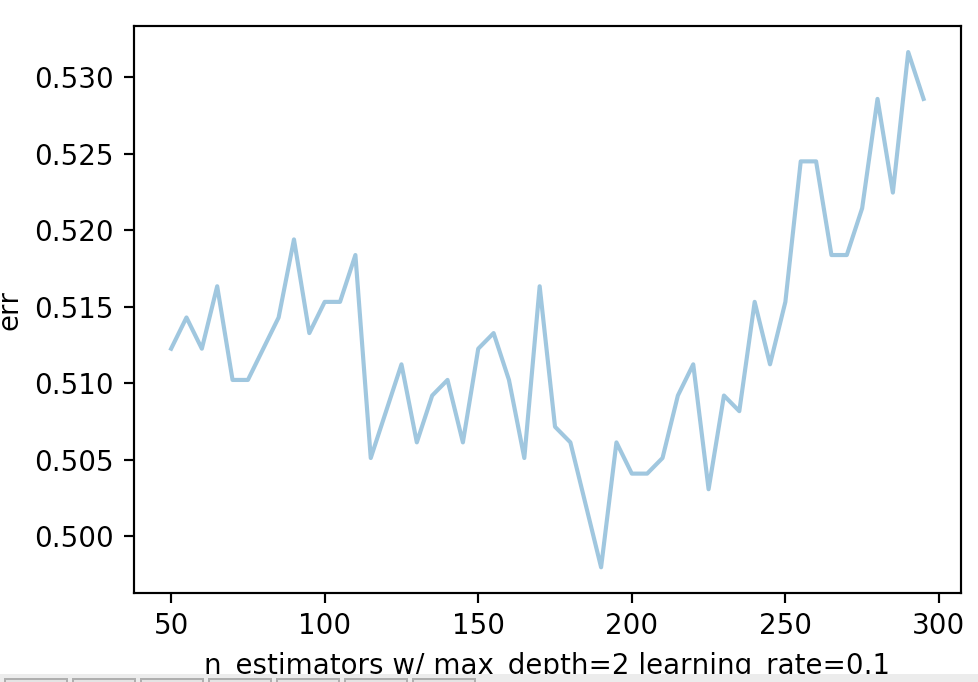
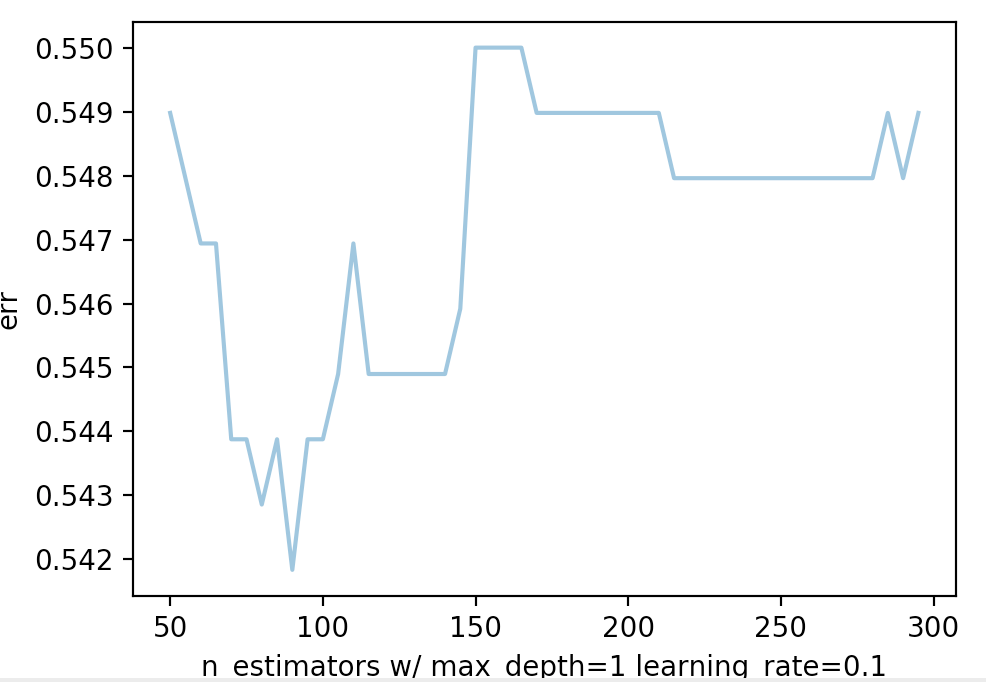
**1 1.00 1.00 1.00 122**

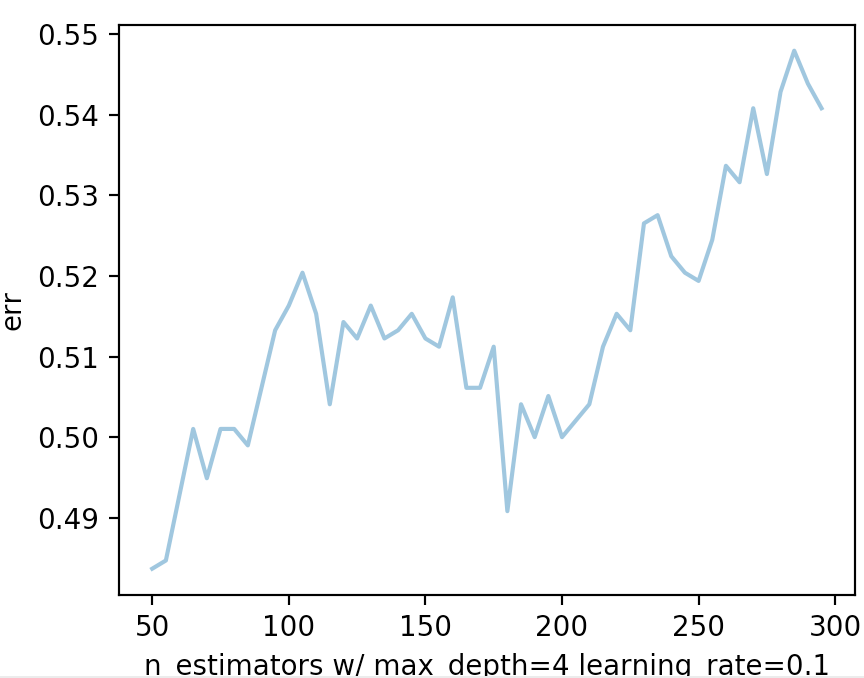
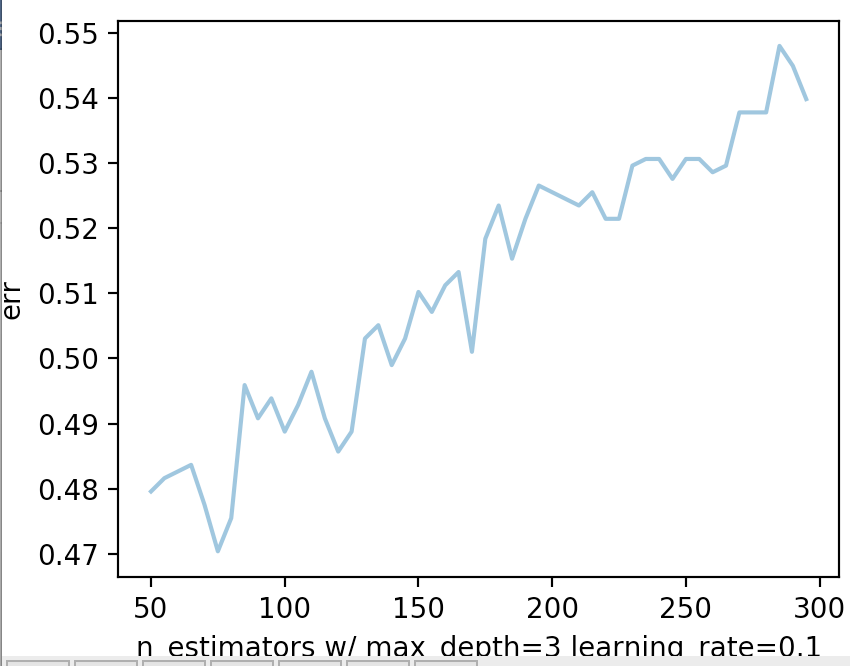
**avg / total 1.00 1.00 1.00 275**

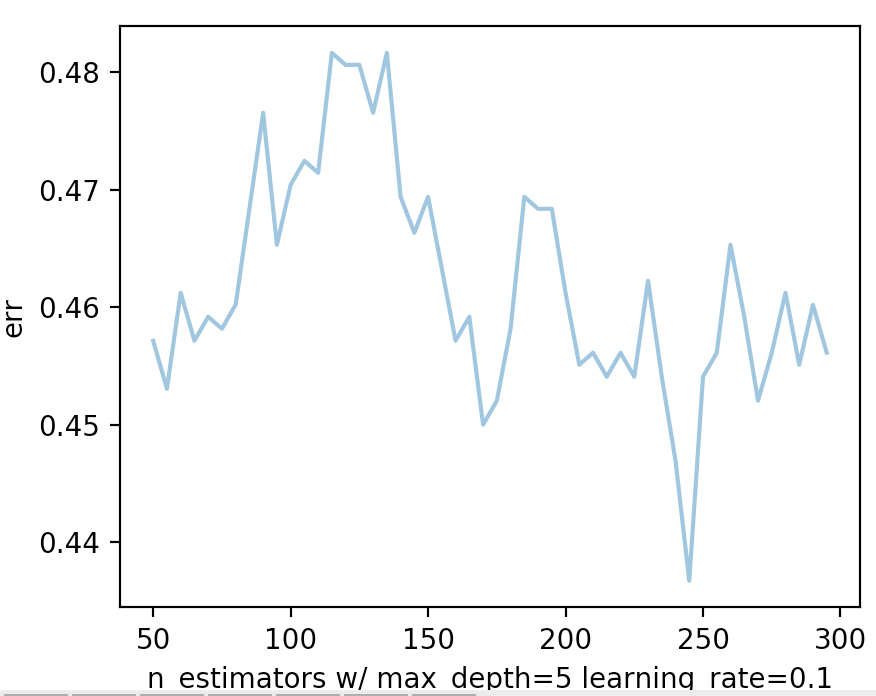
**Boosting - *Wine***

I will be using two classifiers: AdaBoostClassifier and GradientBoostingClassifier.

For AdaBoostClassifier’s base estimator I will use DecisionTreeClassifier, from which the boosted ensemble will be built. Below are the graphs showing the performance on the test data. The learning rate is 0.1.



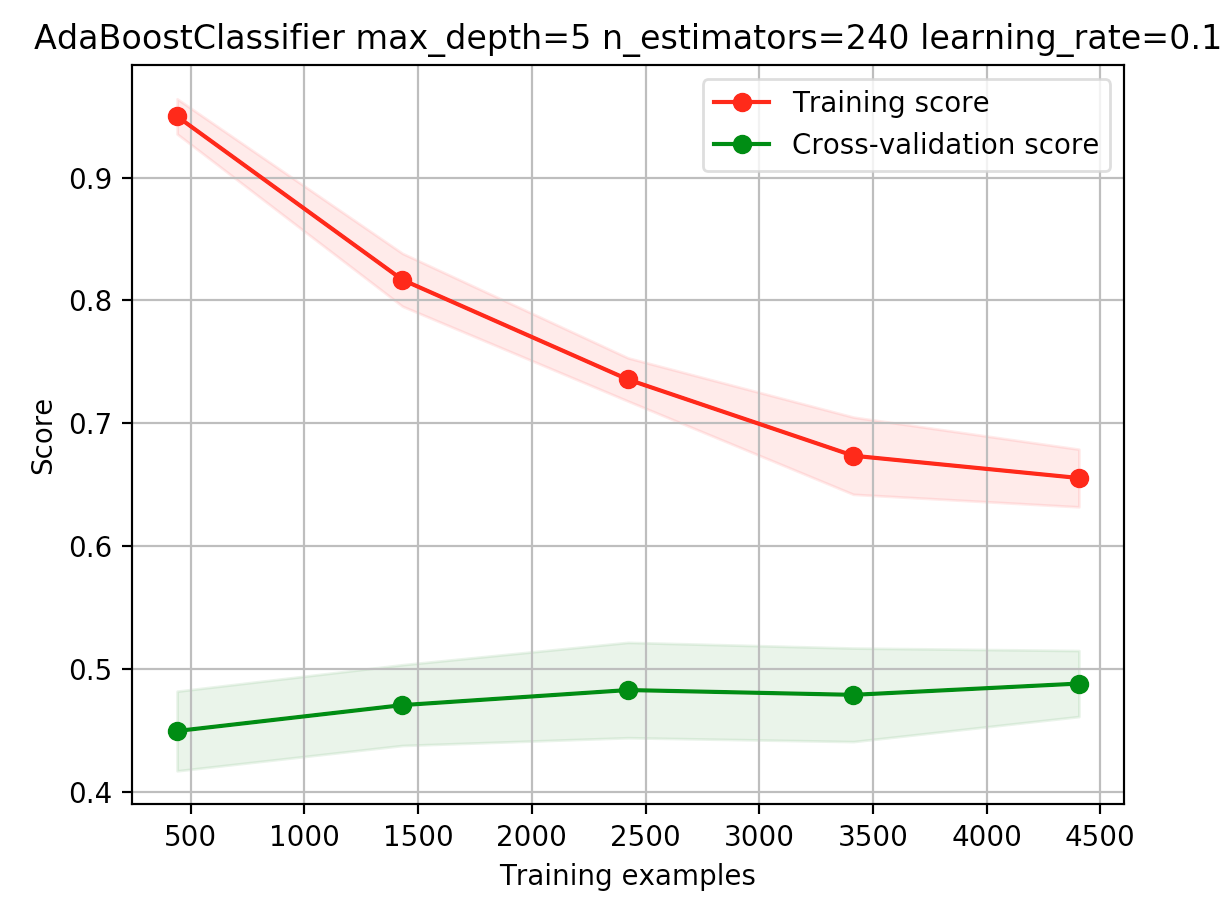




As can be seen from the graphs, AdaBoostClassifier with DecisionTreeClassifier with max\_depth 5 and number of estimators equal to ~240 gives the best accuracy. Let us take a look at a cross-validation score in this case:

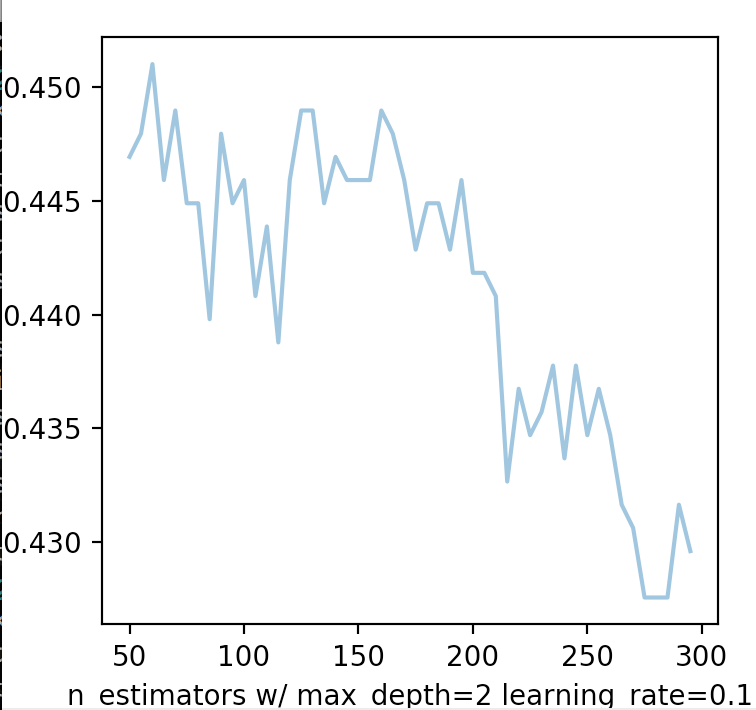
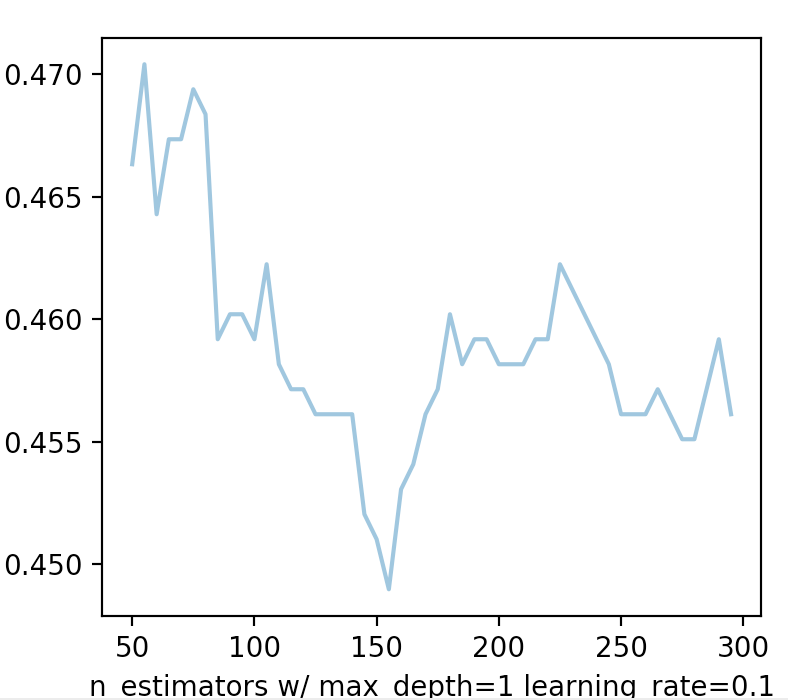
**Mean score: 0.541094077347**

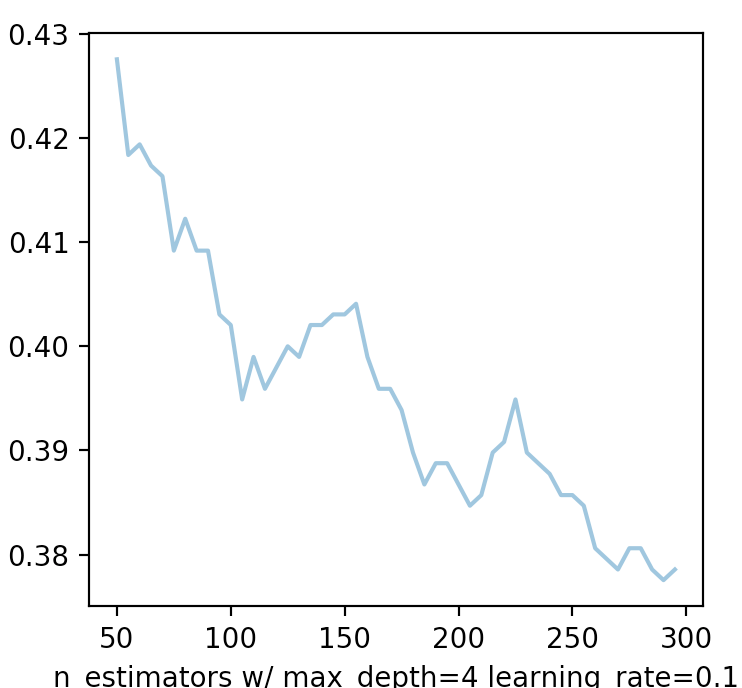
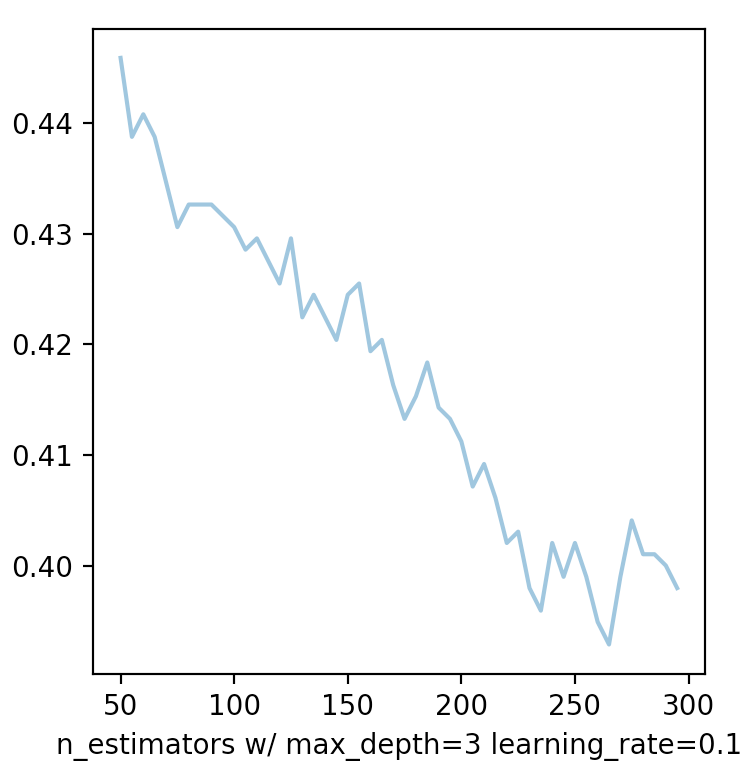
**Standard deviation score: 0.0296215620603**

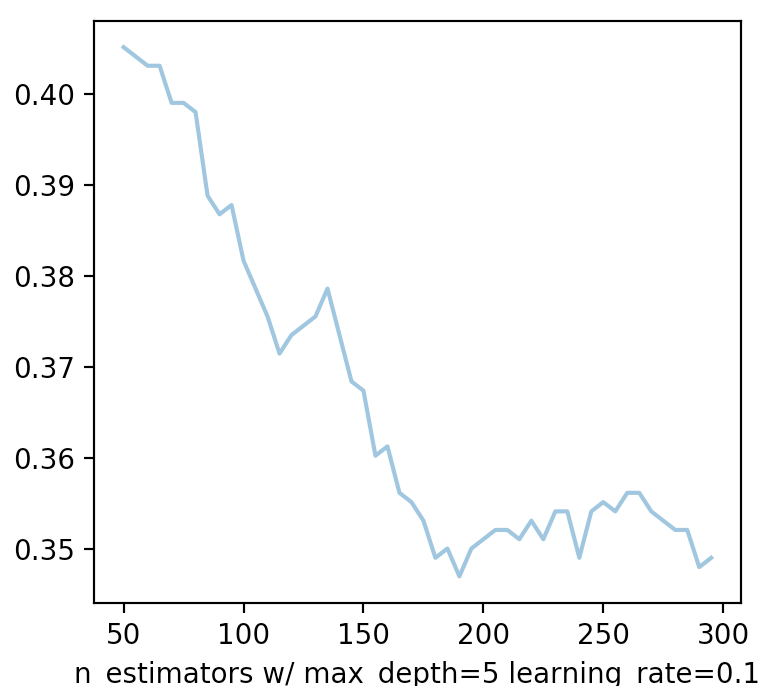


And the accuracy of prediction after running the classifier on the test set is **0.55306122449**.

Let us now try the other classifier – GradientBoostingClassifier. They show the performance on the test data.



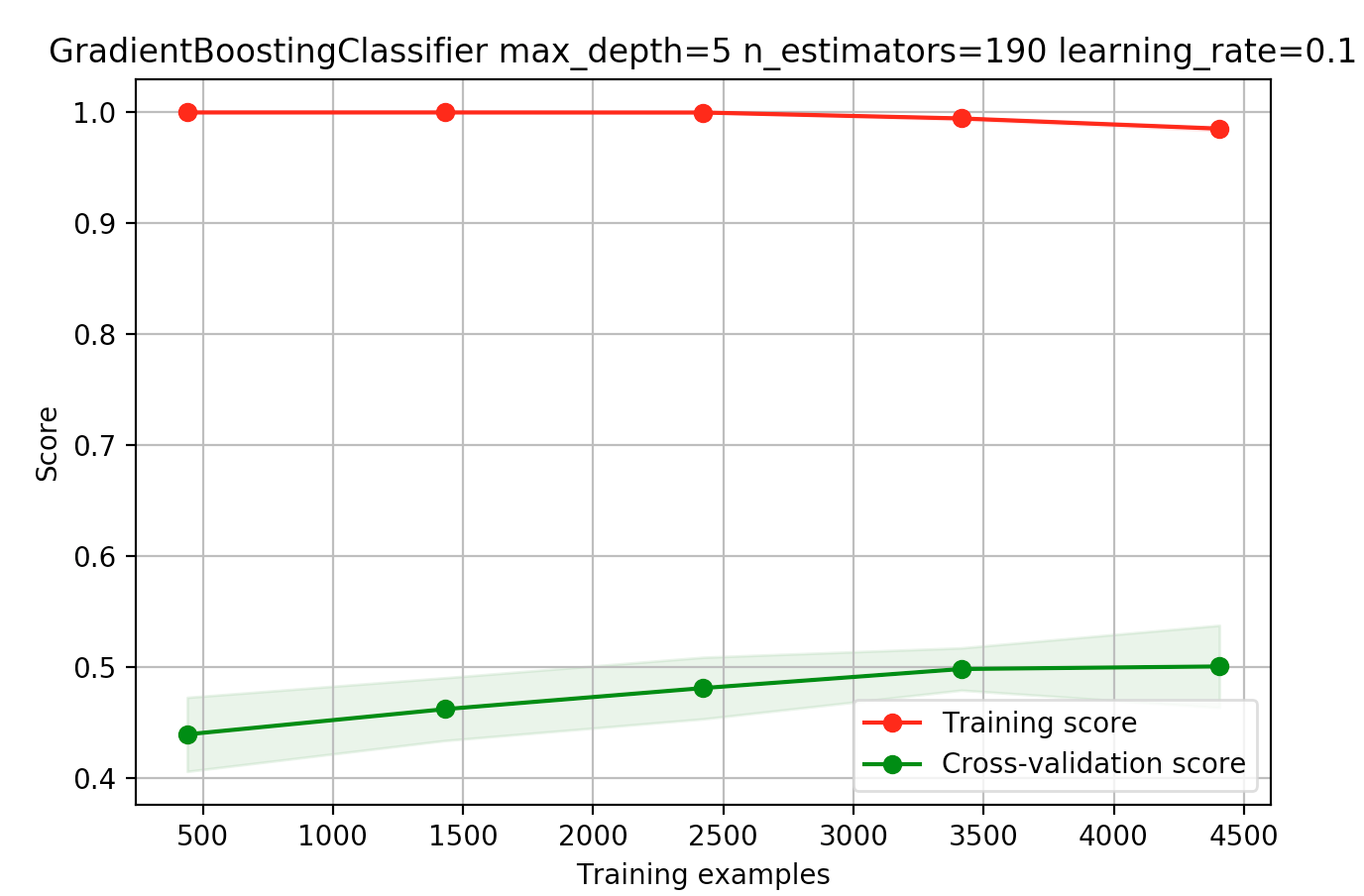




GradientBoostingClassifier definitely performs better than AdaBoostClassifier and much better than DecisionTreeClassifier and MLPClassifier, as can be seen from the cross-validation score:

**Mean score: 0.655962106198**

**Standard deviation score: 0.0150828845708**

****

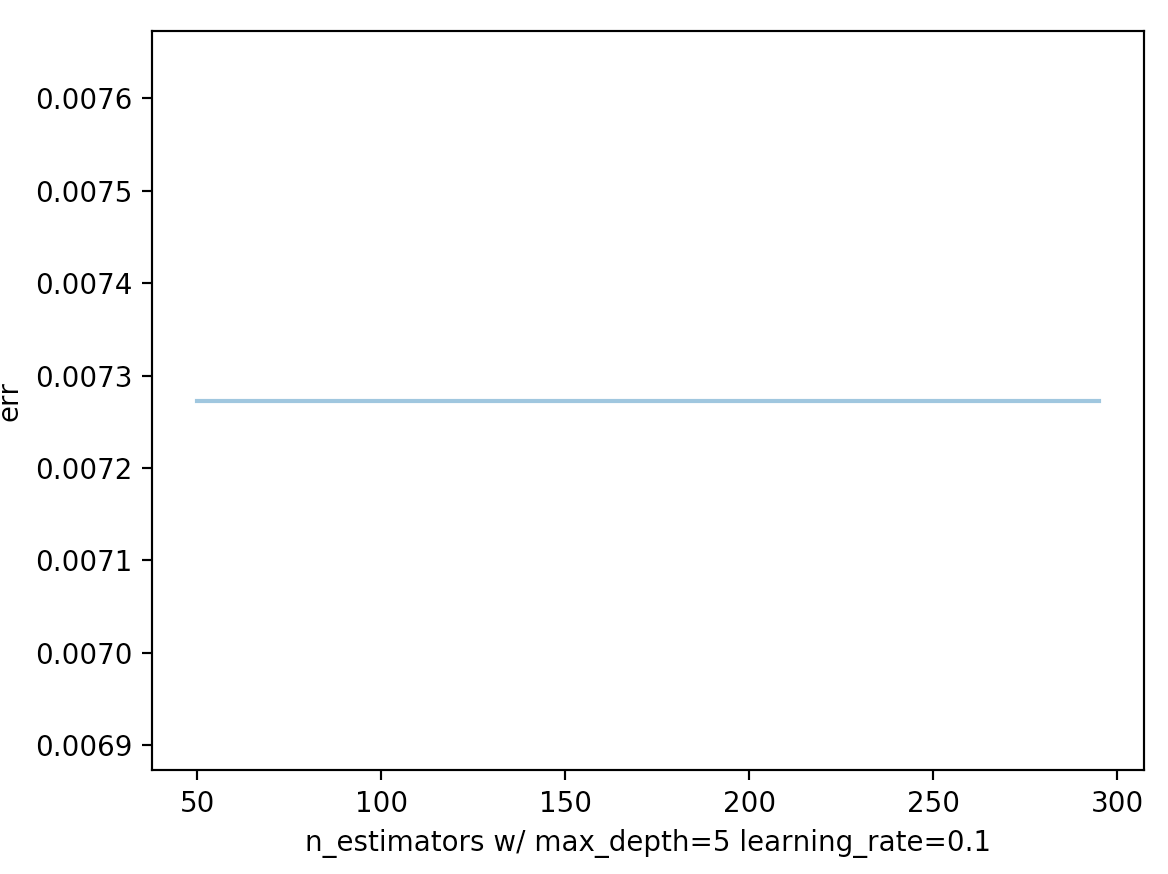
Prediction accuracy on the test set is **0.65306122449**.

In both cases the n\_estimators can be set to a high number. AdaBoostClassifier terminates boosting if perfect fit is found earlier; and GradientBoostingClassifier is robust to overfitting – so a high number should result in better performance.

**Boosting – *Banknote Authentication***

Classifiers: AdaBoostClassifier and GradientBoostingClassifier.

**AdaBoostClassifier**:



10-fold cross-validation of AdaBoostClassifier w/

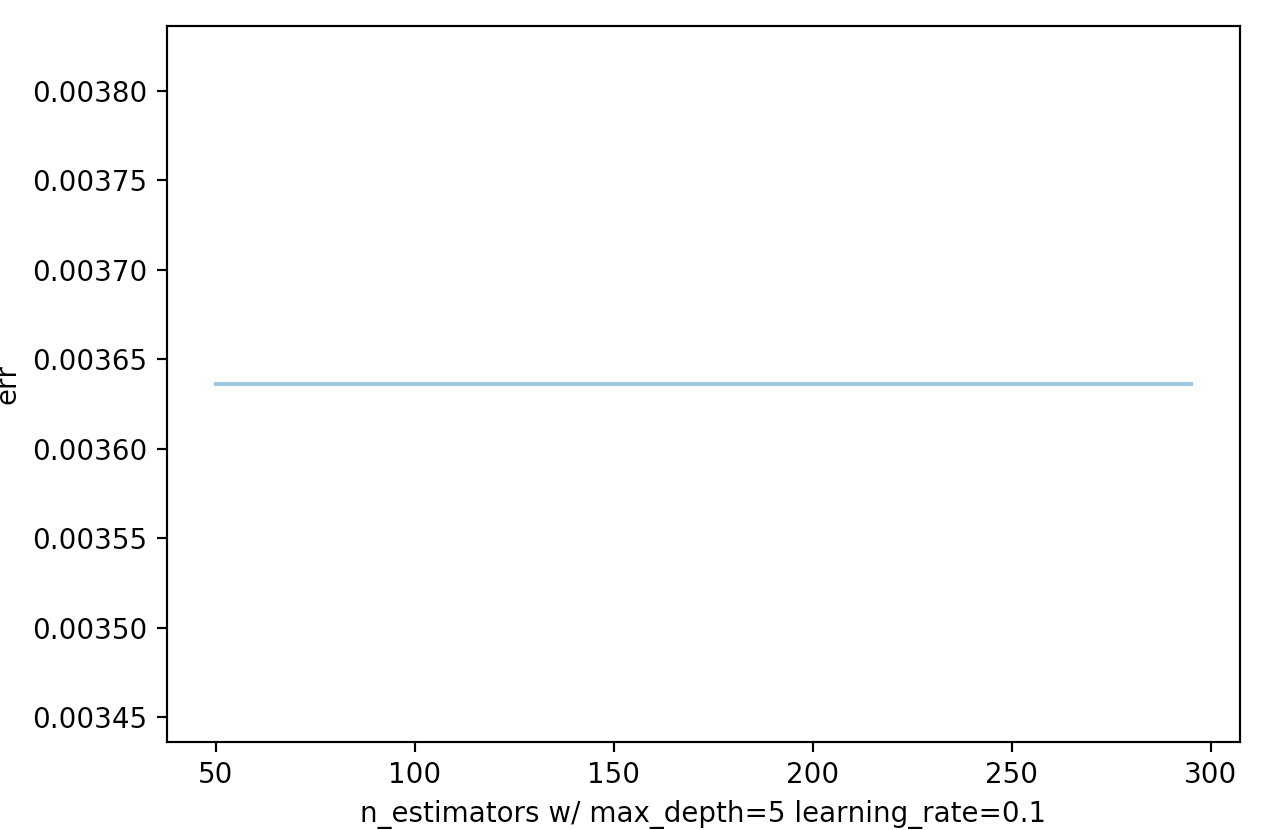
DecisionTreeClassifier(max\_depth=5) as base\_estimator and n\_estimators=240:

Mean score: 0.995404040404

Standard deviation score: 0.00851860169507

Accuracy of prediction on test data: 0.992727272727

**GradientBoostingClassifier**:



10-fold cross-validation GradientBoostingClassifier w/ n\_estimators=190 max\_depth=5:

Mean score: 0.989074228524

Standard deviation score: 0.00892098501166

Accuracy of prediction on test data: 0.996363636364

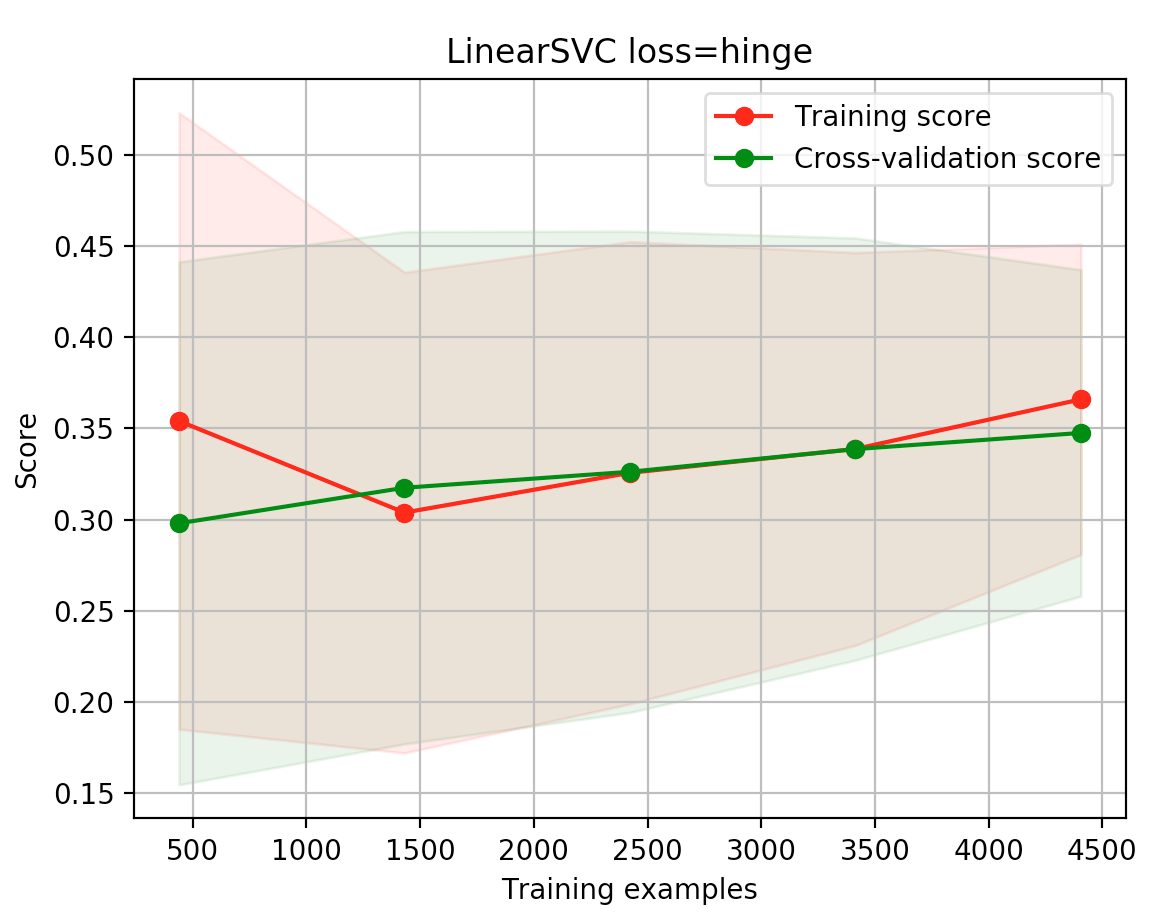
**SVM – *Wine***

**LinearSVC C=1, loss=hinge:**

Cross validation:

10-fold cross-validation mean score: 0.495404901475

10-fold cross-validation standard deviation score: 0.0171675027174



Predictions on test data:

Accuracy of prediction: 0.467346938776

**SVC kernel=poly, degree=3, coef0=1, C=5:**

Cross validation:

10-fold cross-validation mean score: 0.561025504132

10-fold cross-validation standard deviation score: 0.0233703164137

Predictions on test data:

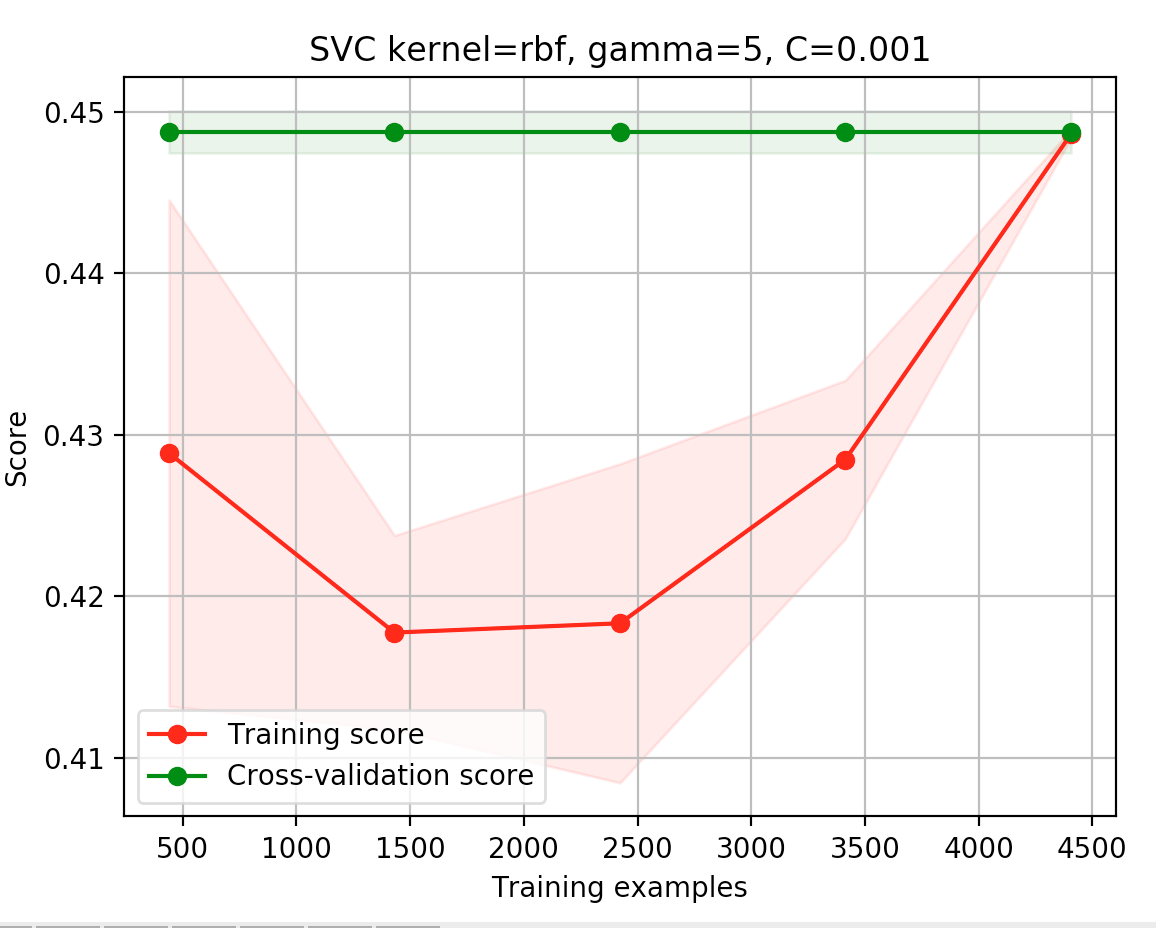
Accuracy of prediction: 0.534693877551

**SVC kernel=rbf, gamma=5, C=0.001**

Cross validation:

10-fold cross-validation mean score: 0.448706824729

10-fold cross-validation standard deviation score: 0.00179517119374



Predictions on test data:

Accuracy of prediction: 0.448979591837

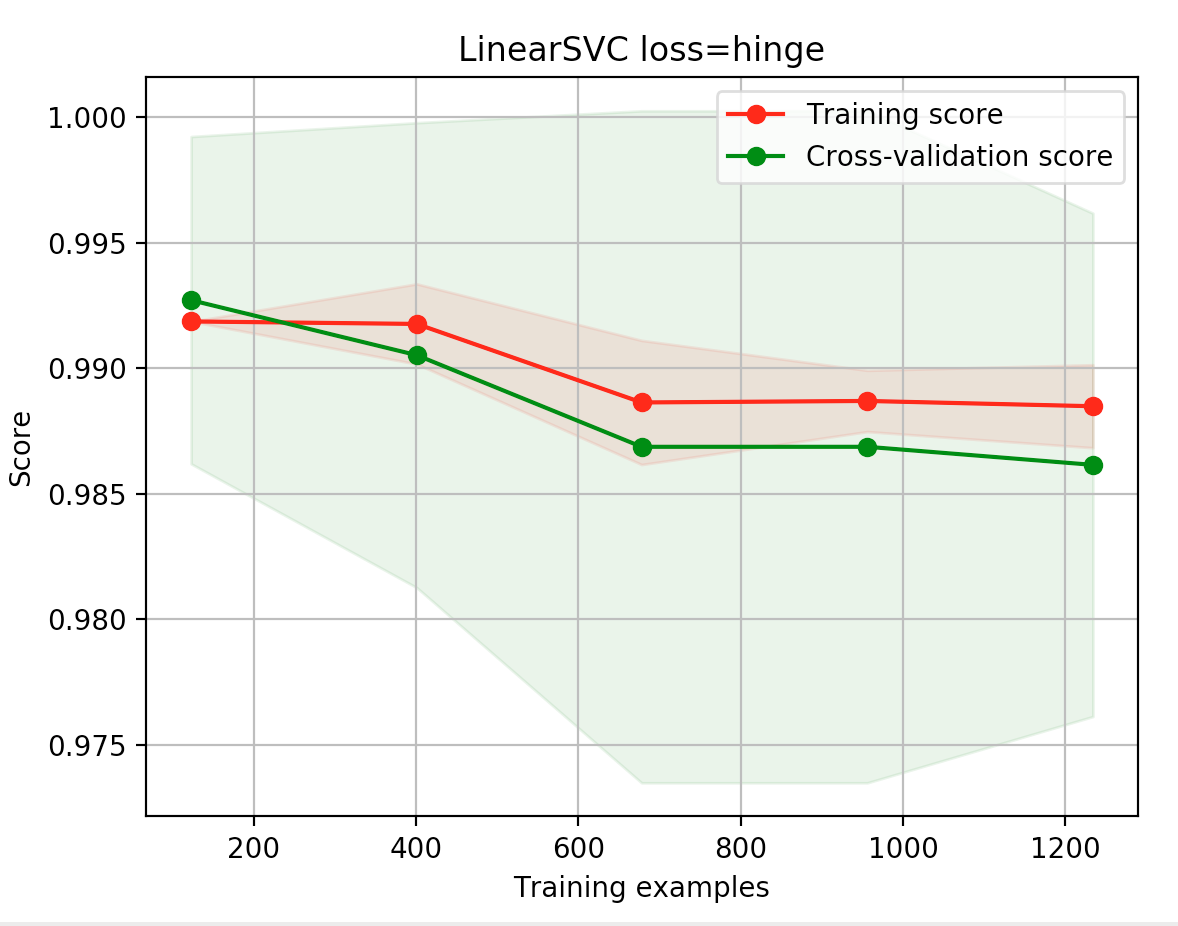
**SVM – *Banknote Authentication***

**LinearSVC C=1, loss=hinge:**

Cross validation:

10-fold cross-validation mean score: 0.984512093411

10-fold cross-validation standard deviation score: 0.0115563379654



Predictions on test data:

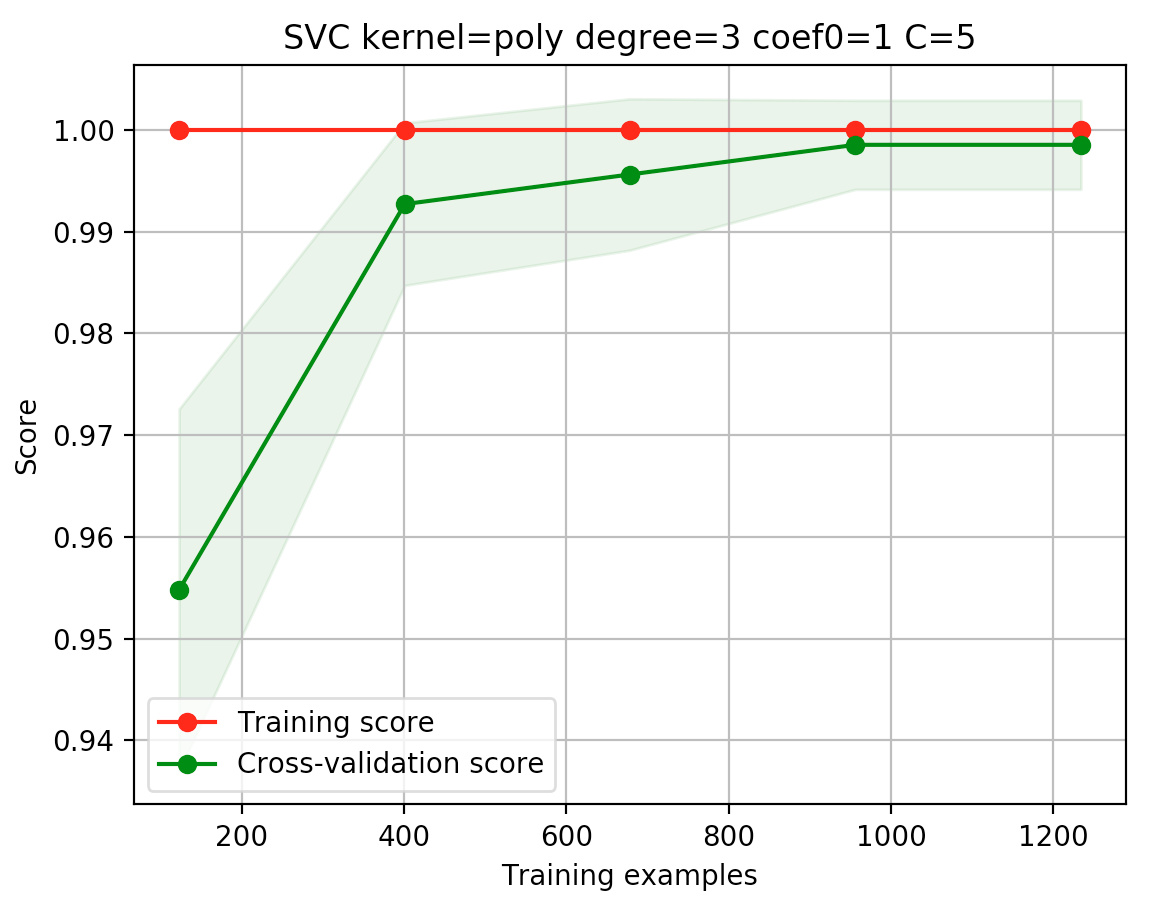
Accuracy of prediction: 0.985454545455

**SVC kernel=poly, degree=3, coef0=1, C=5:**

Cross validation:

10-fold cross-validation mean score: 1.0

10-fold cross-validation standard deviation score: 0.0



Predictions on test data:

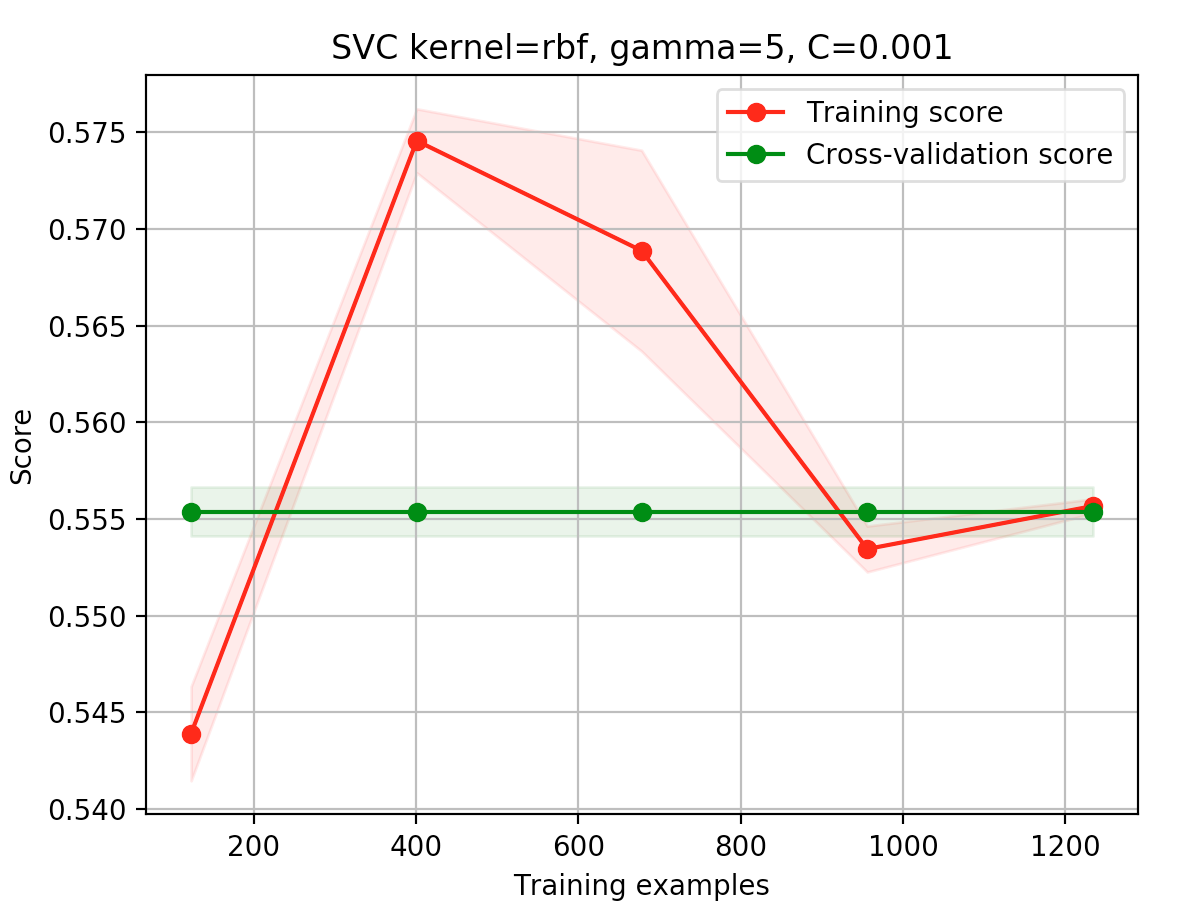
Accuracy of prediction: 1.0

**SVC kernel=rbf, gamma=5, C=0.001:**

Cross validation:

10-fold cross-validation mean score: 0.553319432861

10-fold cross-validation standard deviation score: 0.00187277905799



Predictions on test data:

Accuracy of prediction: 0.563636363636

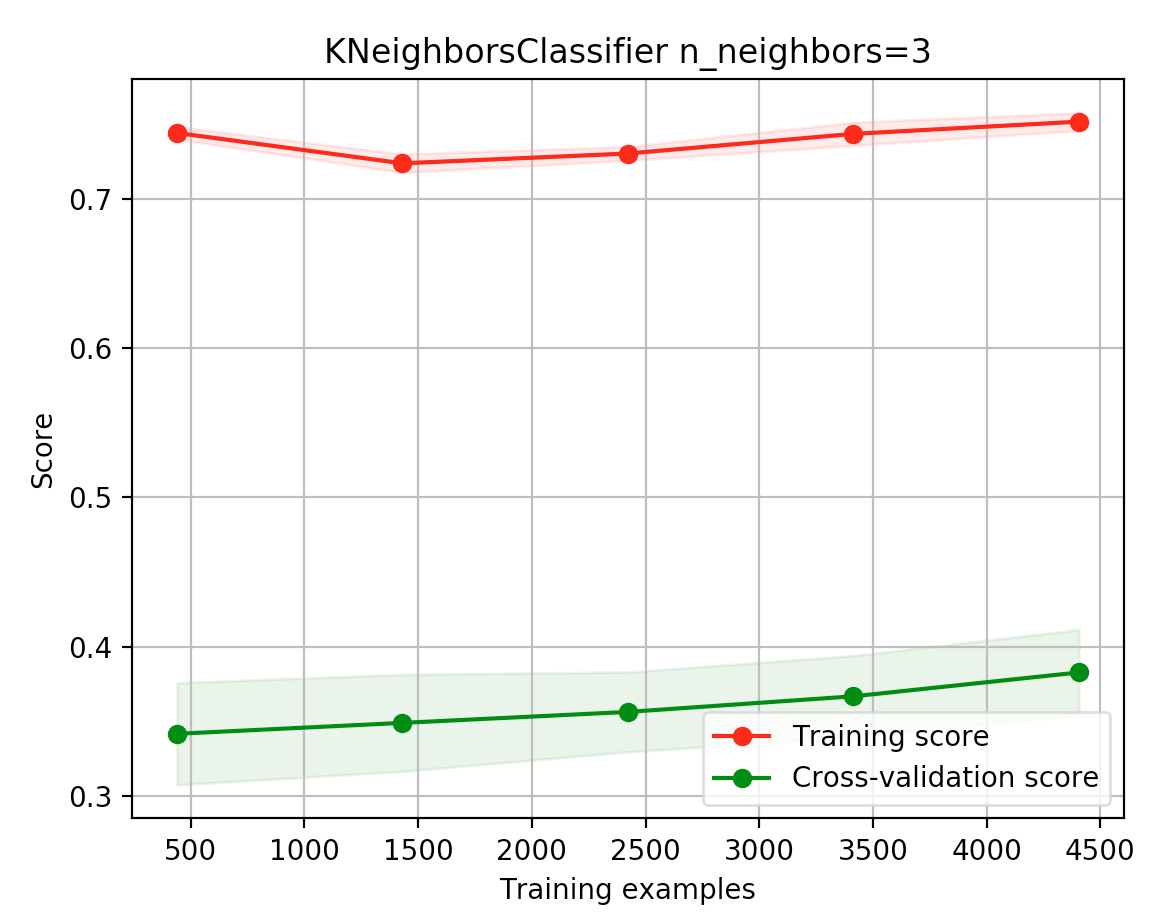
**KNN – *Wine***

**KNeighborsClassifier n\_neighbors=3:**

Cross validation:

10-fold cross-validation mean score: 0.554150209197

10-fold cross-validation standard deviation score: 0.0181072457761



Predictions on test data:

Accuracy of prediction: 0.523469387755

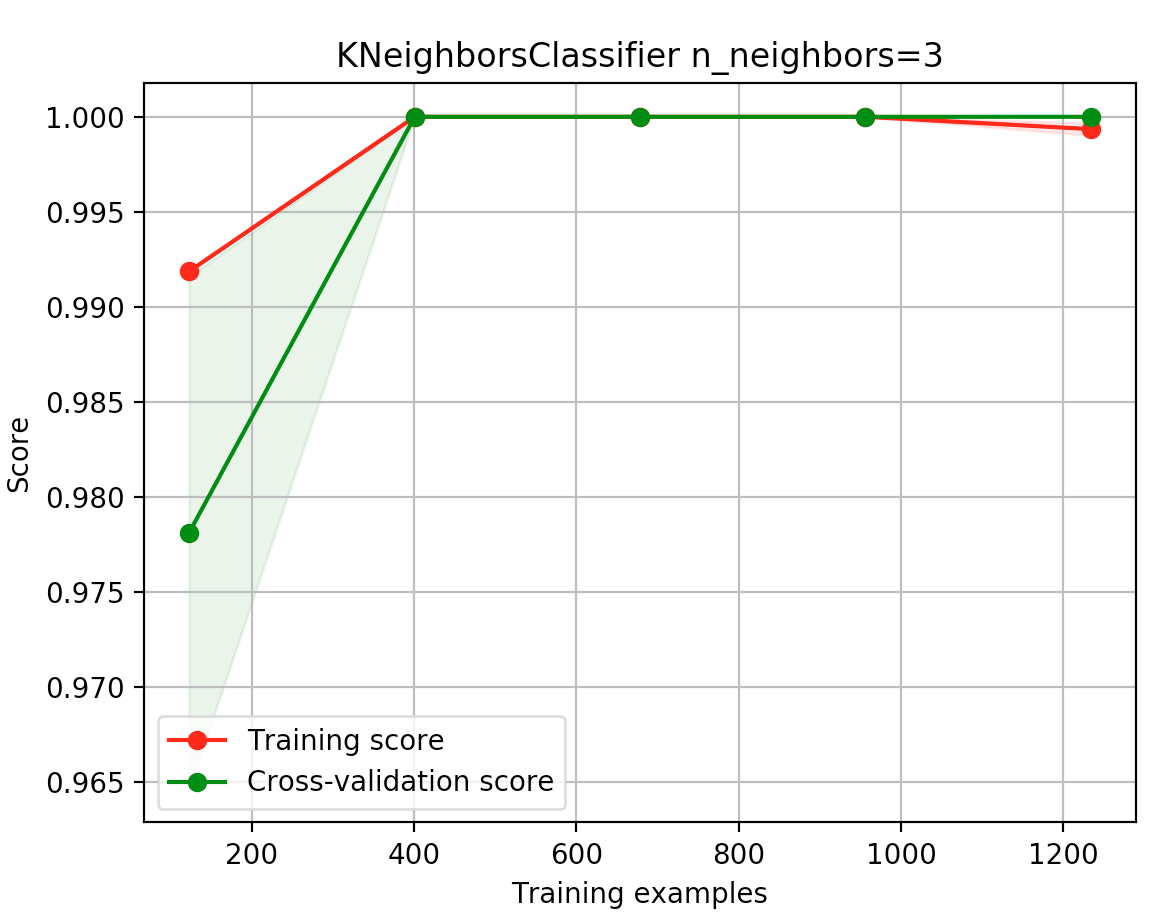
**KNN – *Banknote Authentication***

**KNeighborsClassifier n\_neighbors=3:**

Cross validation:

10-fold cross-validation mean score: 0.998173477898

10-fold cross-validation standard deviation score: 0.00365309180755



Predictions on test data:

Accuracy of prediction: 0.996363636364