Supervised Learning Report

In this report, we applied 5 supervised learning algorithms:

1. Decision Tree (DT),
2. Neural Networks (NN),
3. AdaBoost(AB),
4. Support Vector Machines(SVM),
5. k-Nearest Neighbors(KNN)

on 2 interesting problems (with codes in the following links):

1. Heart Disease Prediction
   https://github.com/yichigo/Heart-Disease-Prediction

2. Classify Gestures by Reading Muscle Activity
   https://github.com/yichigo/Classify-Gestures-by-Reading-Muscle-Activity

1. Heart Disease Prediction

It will be quite useful that heart disease can be predicted from some simple medical measurements. So that people can save a lot of time to treat the disease rather than waiting for the busy doctors and suffering the disease without any treatment.

Dataset:

heart.csv from https://www.kaggle.com/ronitf/heart-disease-uci
there are 13 features:
age, sex, cp (chest pain type), trestbps (resting blood pressure)……
and 1 target with value 0 or 1.
We randomly split the dataset with training size 80% and testing size 20%.

One-Hot Encoding:

For the features which means quantities, we keep the features as they are.

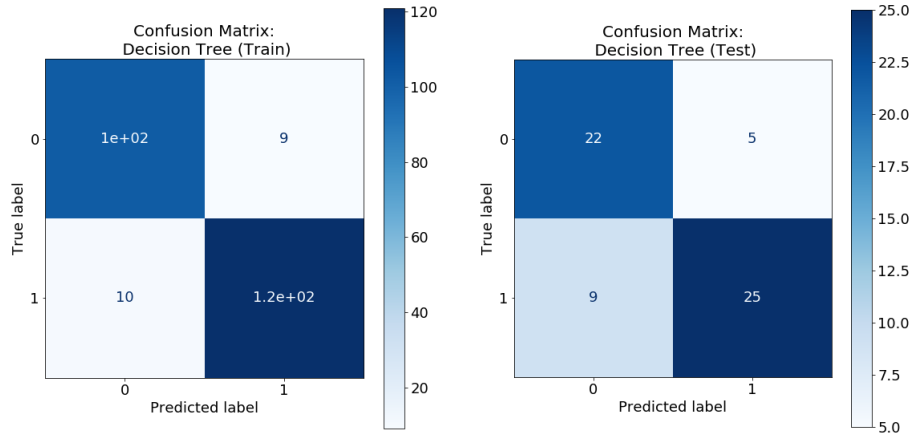
However, for the feature which indicates discrete types, we use one-hot encoding to replace one feature column with multi-columns, and each column represent one type of that feature.

For example, we encoded cp (chest pain type, which could be 0,1,2,3) as cp_1, cp_2, cp_3. To make the features independent, we drop cp_0 since it can be derived from the other 3 types.

1.1 Decision Tree (DT)

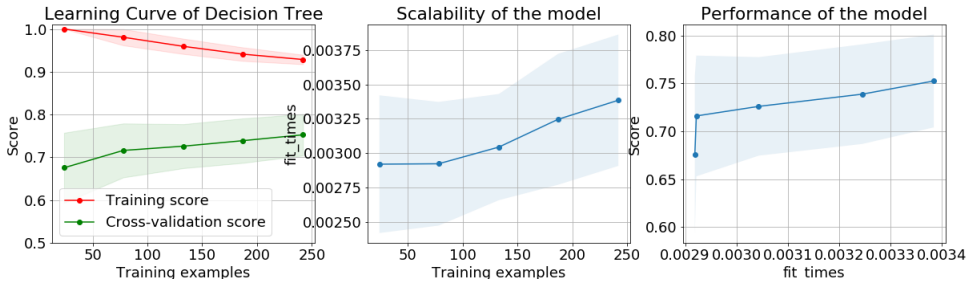
By grid search in [3,5,10,20,50,100], we optimized the max_depth = 5 for the DecisionTreeClassifier.

Confusion Matrix:



The optimized decision tree get a decent accuracy on the testing dataset.

Learning Curve:

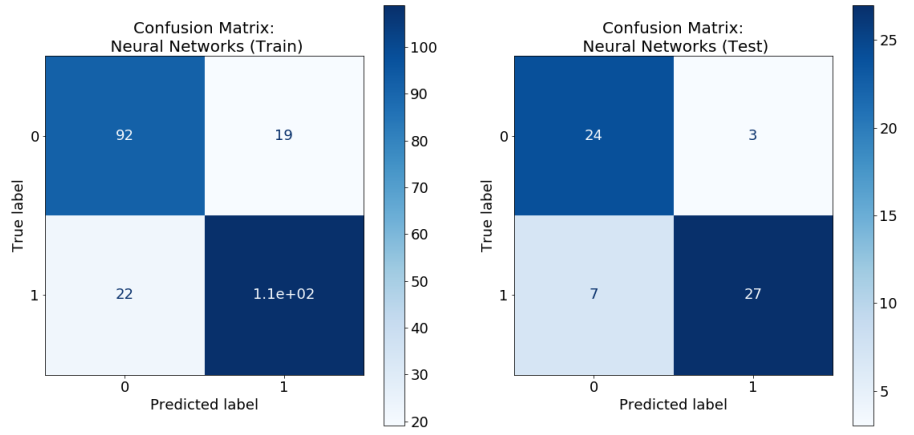


By increasing the training size, the training score decreases and the cross-validation score increases to about 0.75. In addition, DT is very fast to fit the data.

1.2 Neural Networks (NN)

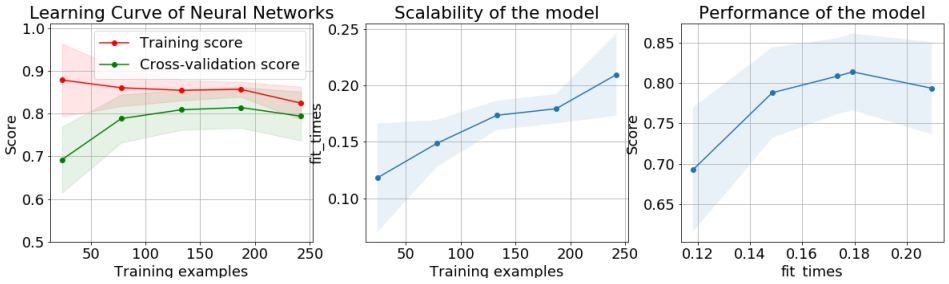
By grid search, we optimized the hidden_layer_sizes = (20, 20), activation function = 'identity', solver = 'adam' and max_iter = 200 for the 2-layer Neural Networks.

Confusion Matrix:



The testing performance of Neural Networks is quite good and very close to the performance on the training dataset.

Learning Curve:



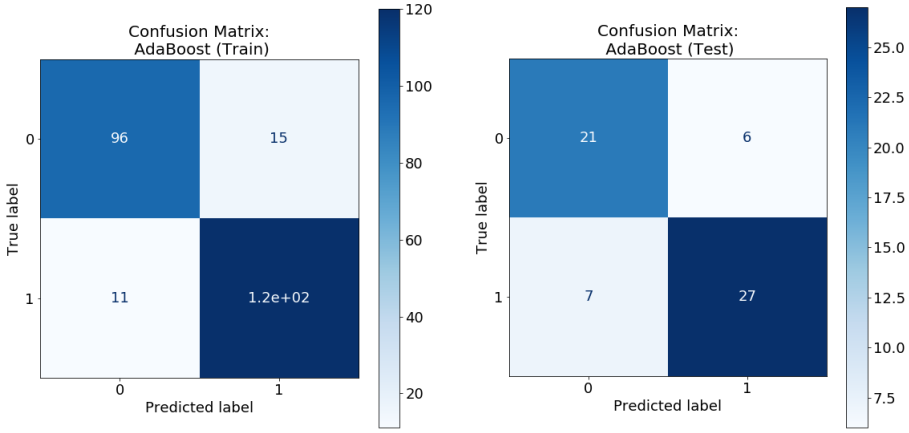
From the learning curve, we can see that by increasing the training size, the training score decreases and the cross-validation score increases. And the optimized region has very similar scores. It shows the optimized model has quite less overfitting.

NN is the best performed model in this problem, however, it is about 100 times slower than DT.

1.3 AdaBoost(AB)

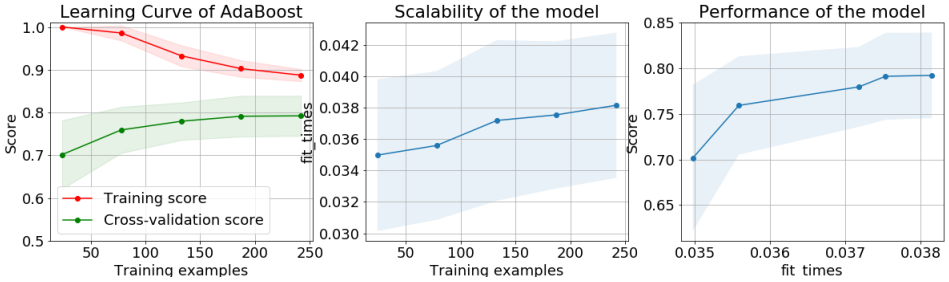
By grid search, we optimized the n_estimators =20.

Confusion Matrix:



AdaBoost performed good on both training dataset and testing dataset.

Learning Curve:



The learning curve is similar with DT. The optimized performance is about 0.8, which is better than DT but a little bit worse than NN.

The speed of Adaboost is also in the middle, the running time is about 10 times of DT, and 1/10 of NN.

1.4 Support Vector Machines(SVM)

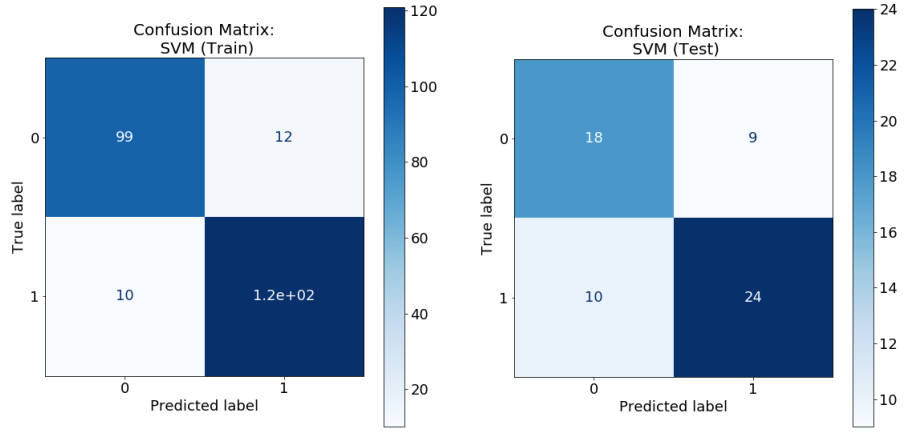
For the kernel function, we tried both 'rbf' and 'sigmoid', where

$K(X,Y)=\exp(\|X-Y\|^2/2\sigma^2)$ for 'rbf'

$K(X,Y)=\tanh(\gamma \cdot X^T Y+r)$ for 'sigmoid'

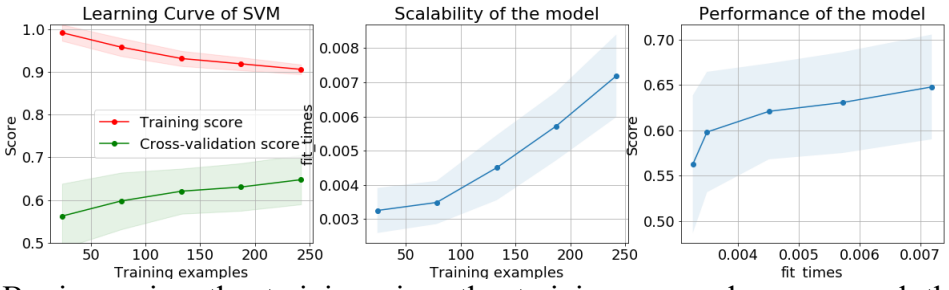
the optimized model gave us 'rbf' rather than 'sigmoid'. And we also optimized the C = 10 and gamma =0.001 by using grid search.

Confusion Matrix:



The testing performance is not as good as the previous 3 models.

Learning Curve:

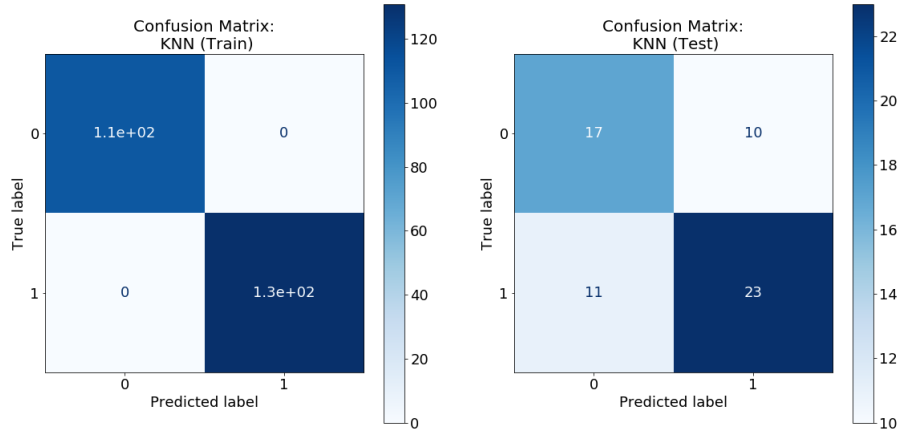


By increasing the training size, the training score decreases and the cross-validation score increases to about 0.65. SVM has worse performance than other previous models. The speed is fast, only slower than DT, since the size of the data is quite small.

1.5 k-Nearest Neighbors(KNN)

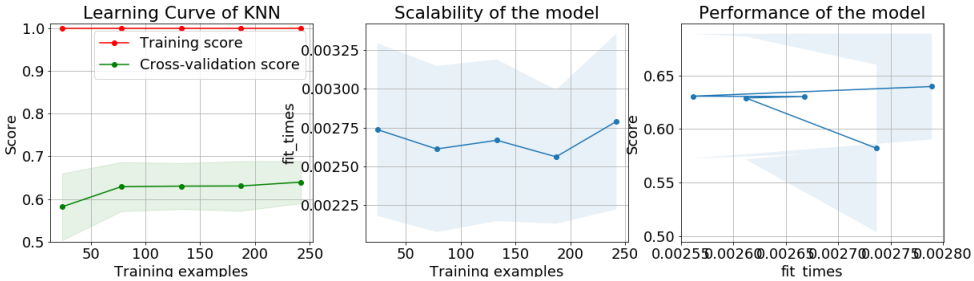
By grid search, we optimized the n_neighbors = 10, and weights = 'distance'

Confusion Matrix:



The testing performance is about similar with SVM classifier.

Learning Curve:



From the learning curve, we can find that although the optimized performance is about the same with SVM (0.65), the kNN model get the optimized performance on less than half of the total data size. In addition, the speed is even faster than DT.

Conclusion

NN is the best, and kNN is the fastest. To balance the performance and the fitting speed, we recommend DT and AdaBoost. Forget about SVM in this problem.

2. Classify Gestures by Reading Muscle Activity

If the machine learning algorithms can figure out people's gestures, then this powerful tool can be applied in many areas, such as recoding sign language, taking care of children or patients, etc.

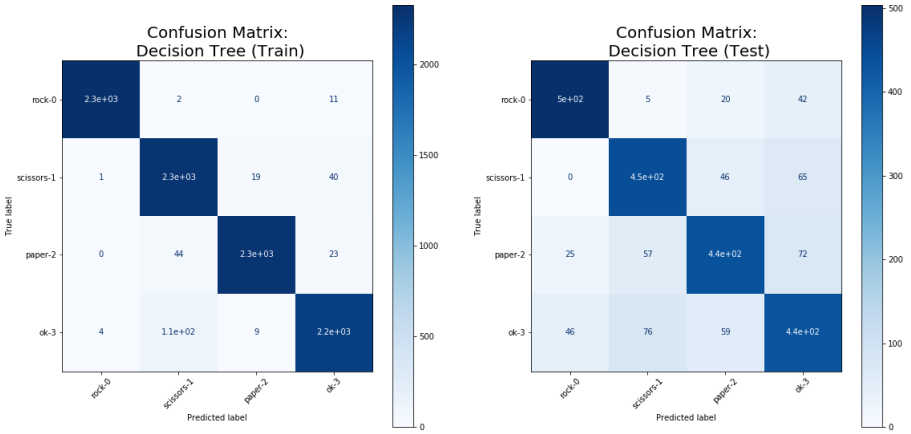
Dataset:

emg-4/0-3.csv from https://www.kaggle.com/kyr7plus/emg-4
there are 8 x 8 = 64 features:
    there are 8 muscle readings, and each one has 8 sensors.
and 1 target with 4 gesture types:
    rock - 0, scissors - 1, paper - 2, ok - 3.
We randomly split the dataset with training size 80% and testing size 20%.

2.1 Decision Tree (DT)

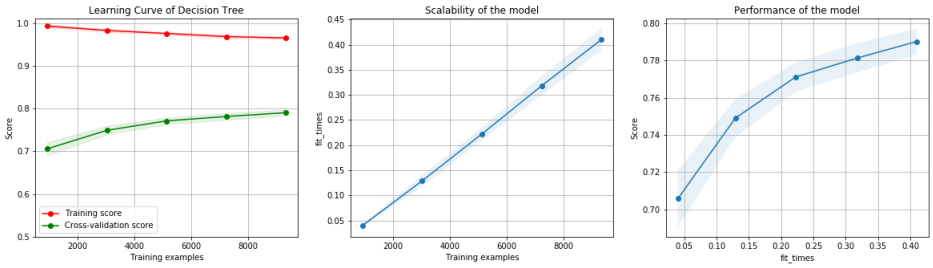
By grid search in [10,20,50], we optimized the max_depth = 20 for the DecisionTreeClassifier.

Confusion Matrix:



The testing proformance is good, but obviously worse than training performance.

Learning Curve:

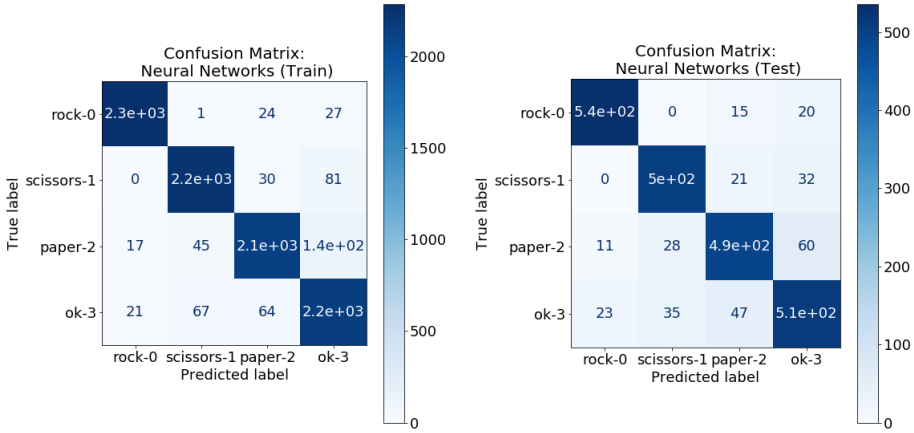


By increasing the training size, the training score does not decrease too much, and the cross-validation score increases to about 0.8. The speed of DT is still fast on larger size of dataset.

2.2 Neural Networks (NN)

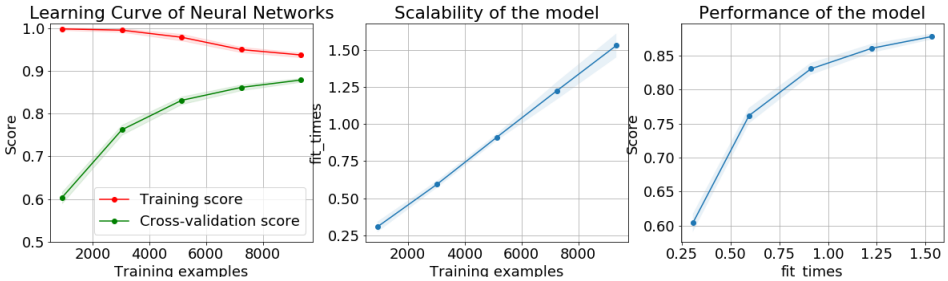
By grid search, we optimized the hidden_layer_sizes = (25, 15), activation function = ‘relu, solver = ‘lbfgs’ and max_iter = 100 for the 2-layer Neural Networks.

Confusion Matrix:



The performance of NN is the best on testing dataset.

Learning Curve:

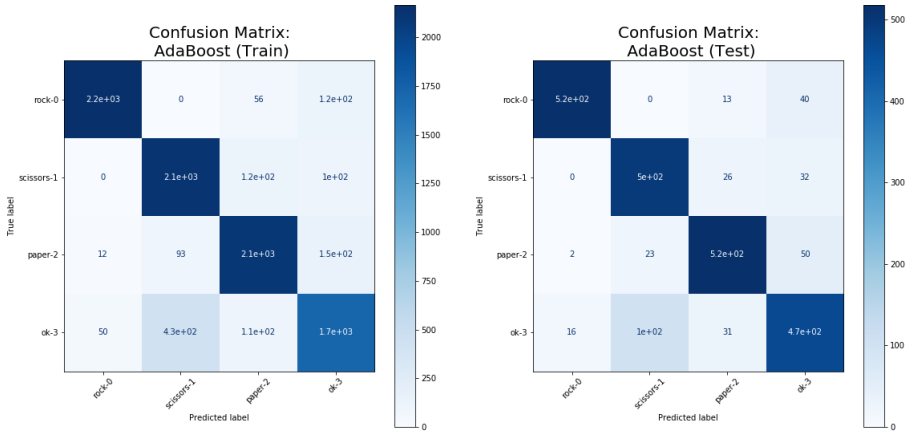


The cross-validation score increased to about 0.9 in the learning curve. NN is the best performed model in this problem. Very small overfitting. The speed is still good, the running time is about 3 times of DT and AdaBoost.

2.3 AdaBoost(AB)

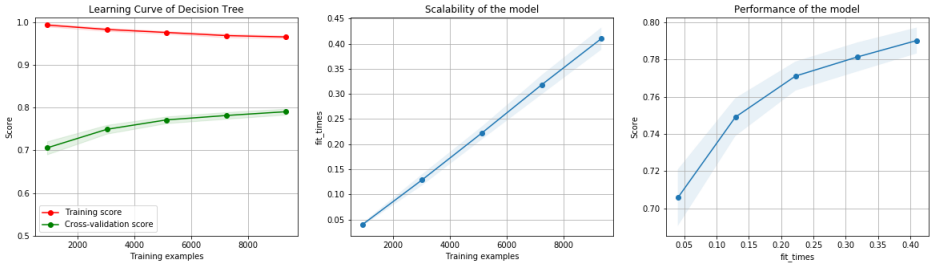
We optimized n_estimators =100 by grid search.

Confusion Matrix:



The Performance looks even better than DT.

Learning Curve:



The learning curve is similar with DT. And the speed is also similar with DT.

2.4 Support Vector Machines(SVM)

Although we optimized the hyper-parameters (c value, gamma value and kernel) by grid search, the performance is still bad in the testing dataset.

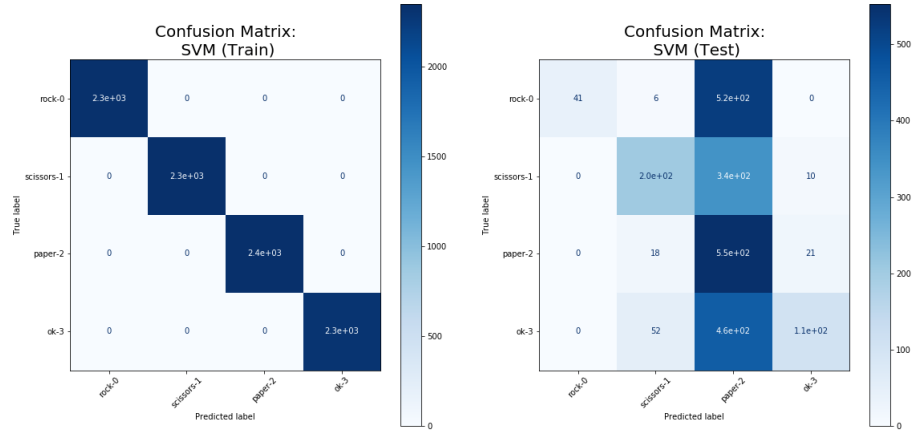
For the kernel function, we tried both 'rbf' and 'sigmoid', where

$K(X,Y)=\exp(\|X-Y\|^2/2\sigma^2)$ for 'rbf'

$K(X,Y)=\tanh(\gamma \cdot X^T Y+r)$ for 'sigmoid'

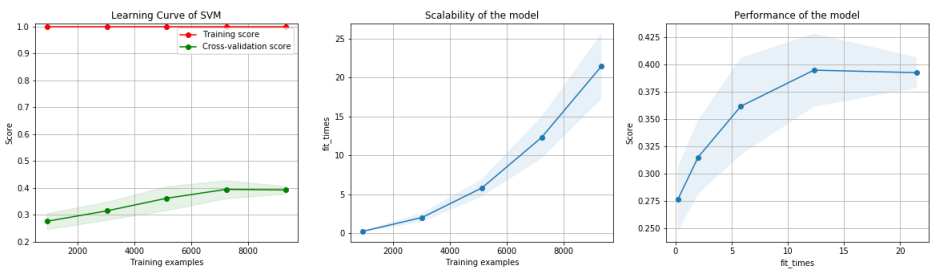
the optimized model below use 'rbf'. And 'sigmoid' is even worse than 'rbf' in this problem.

Confusion Matrix:



SVM overfit the data a lot, the performance is about perfect on the traning dataset, but too much bad on the testing dataset.

Learning Curve:



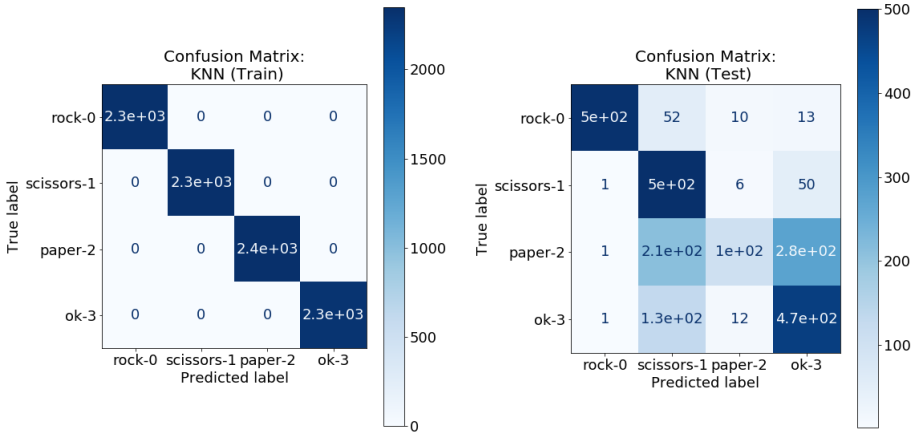
From the learning curve, we can see that the cross-validation score is much lower than the training score. The highest value is only 0.4.

In addition, the speed of SVM is too slow in larger dataset. About 100 slower than DT and AdaBoost.

2.5 k-Nearest Neighbors(KNN)

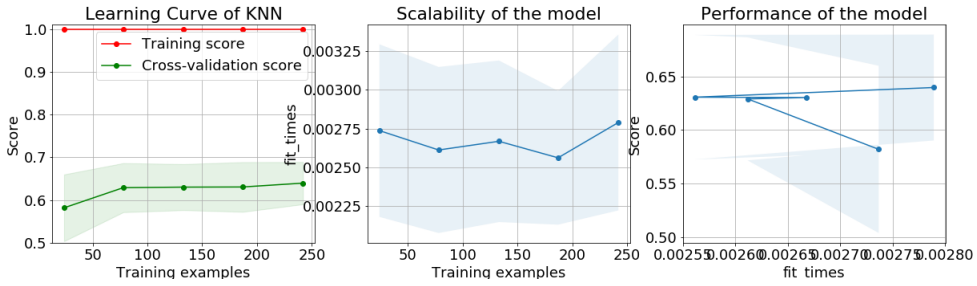
By grid search, we optimized the n_neighbors = 5, and weights = 'distance'

Confusion Matrix:



The testing performance is bad. But better than SVM.

Learning Curve:



The cross-validation score does not increase too much in the learning curve. It is fast, but bad.

Conclusion

To balance the performance and the fitting speed, NN is the best choice. We also recommend DT and AdaBoost, they are a little worse than NN but a little faster than NN. Forget about SVM and KNN in this problem.