# Problem 1

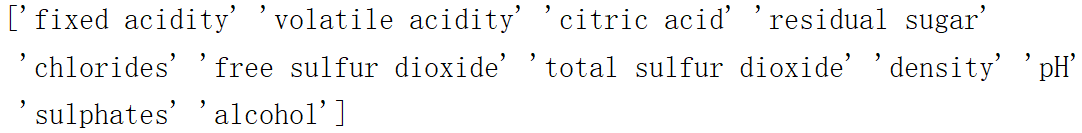
## Problem statement

### Dataset

The dataset is red variant of the Portuguese "Vinho Verde" wine appraisal samples. The goal is to model wine quality based on physicochemical tests.



11 Features:

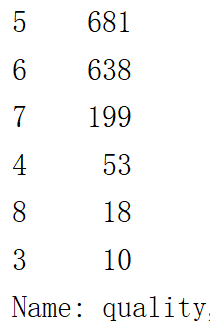
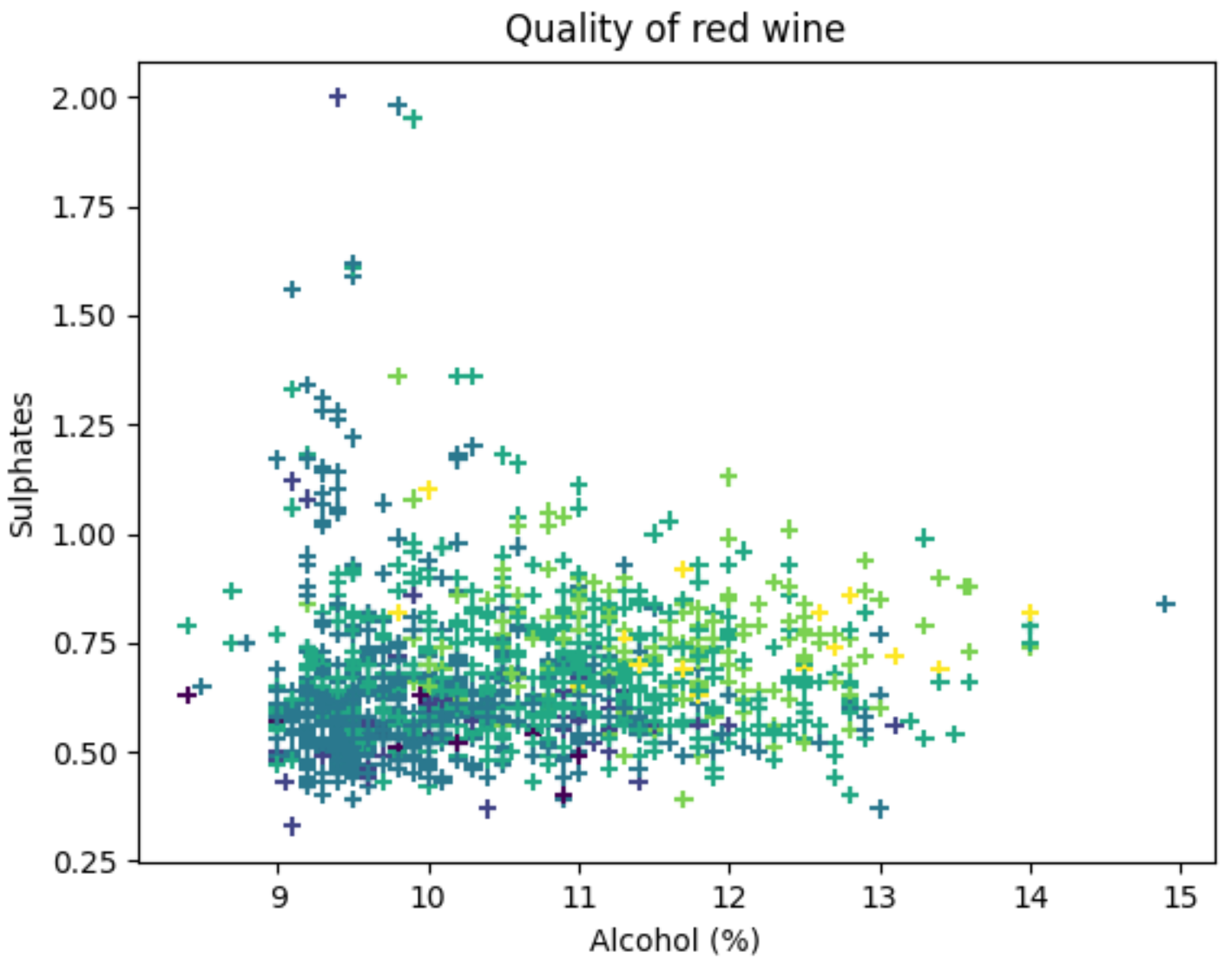


Target: quality (score between 0 and 10), only 3 to 8 samples in this dataset

What interests me is that this dataset can be viewed as both multi-classification and binary-classification problems in order to satisfy different requirements. I might get some interesting findings comparing the results of this two problems. Moreover, the classes are not balanced, there are much more normal wines than excellent or poor ones. I’m curious will or how this unbalance affects the result.

Problem 1 is the multi-classification model. I’ll do the latter in Problem 2.

### Visualize

I used two dominant features, which I got from decision tree, to visualize the dataset. The boundaries are not clear. Obviously, it will be a tough task.

### Split training / test set

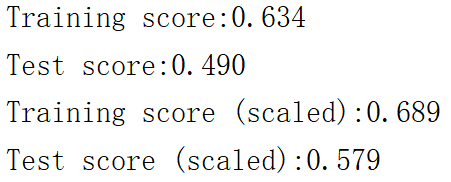
I randomly split the dataset with 0.3 of it as test set and keep the same distribution in both sets.

## k-Nearest Neighbors

### Default model

I don’t know exactly which algorithm will benefit from scaling. So I decided to test it myself.

I used the default model (parameters) to fit my training set at first:

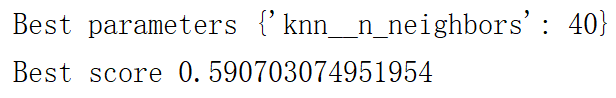


From the results above, I take the answer is yes for KNN

### Tune hyper-parameters

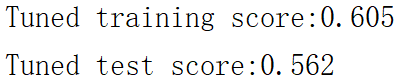
I used the Grid search cross validation to find the optimal k:





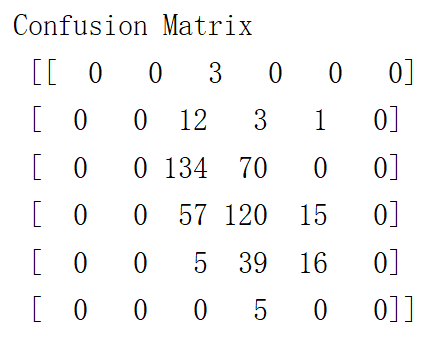
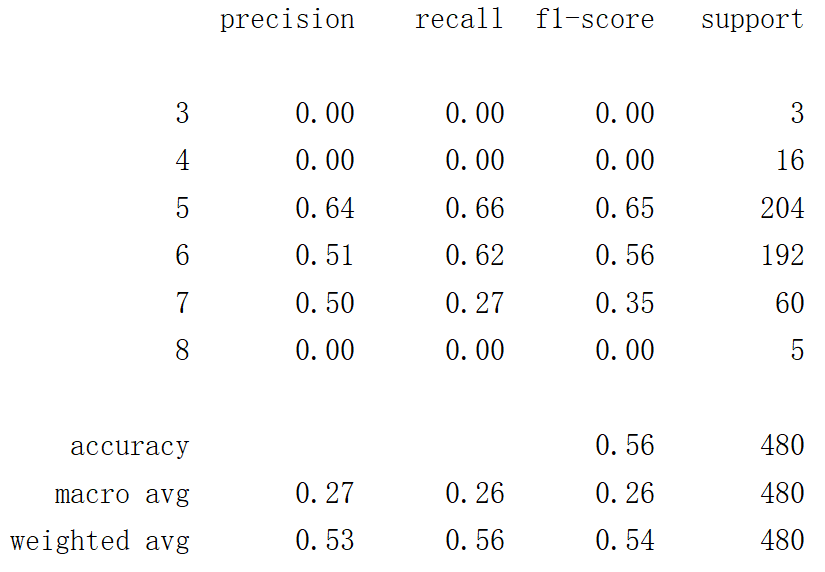
### Optimal model

1. Predict



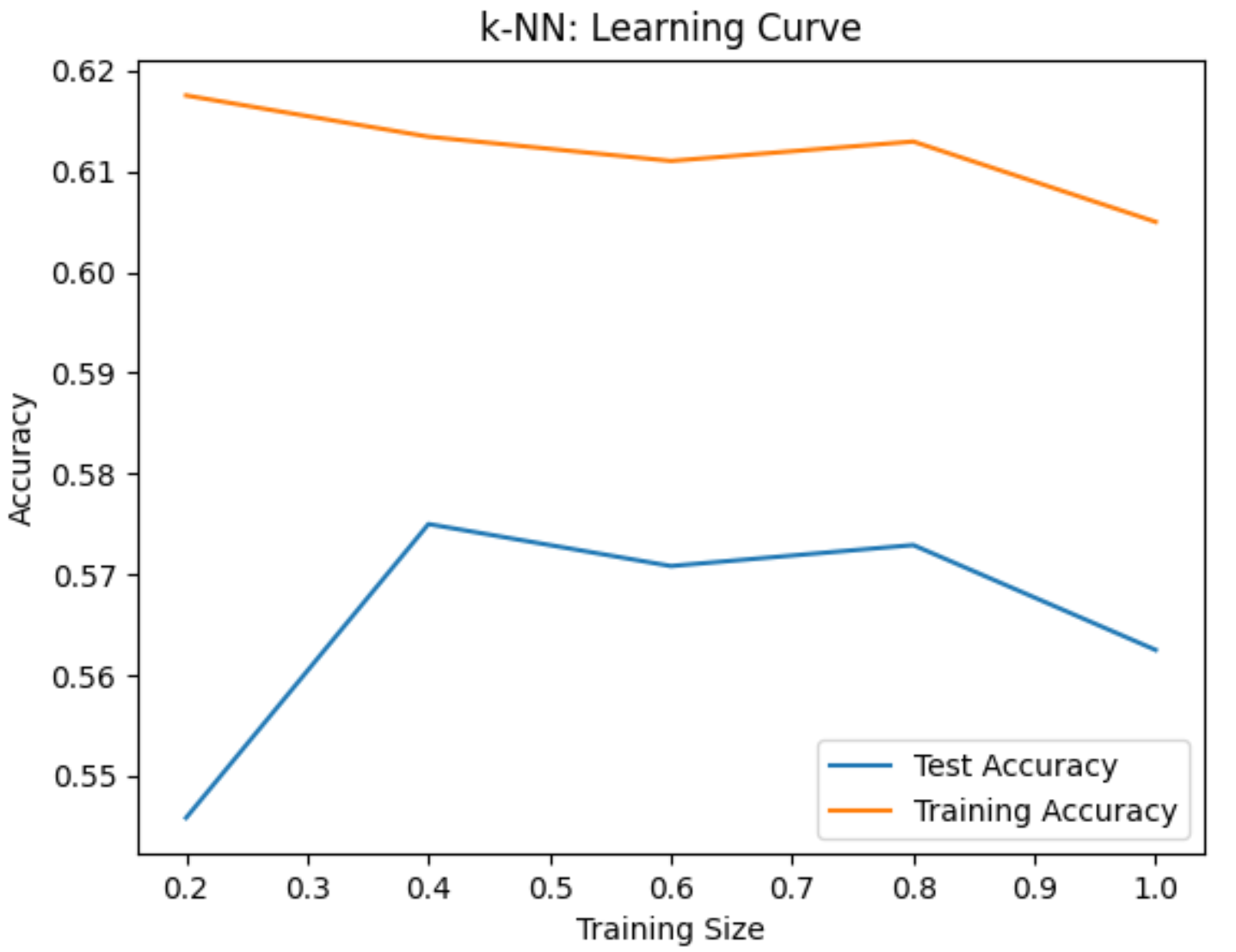
Using the optimal parameter, I got the training/test score above. Since the k-40 is larger than default, it is normal to see lower training score. While test score getting lower too, it is a sign of underfitting.

1. Confusion matrix

The reason why KNN performed badly is that class 3, 4, 8 don’t have sufficient data (training set has the same distribution as test set). And this is the prove to support unbalaced classification indeed affects the result, espacialy for KNN.

1. Learning curve

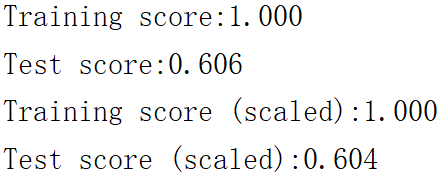


The X axis is the portion of training set I used to plot the learning curve.

It might because of the underfitting that test accuracy did not benefits from larger training set.

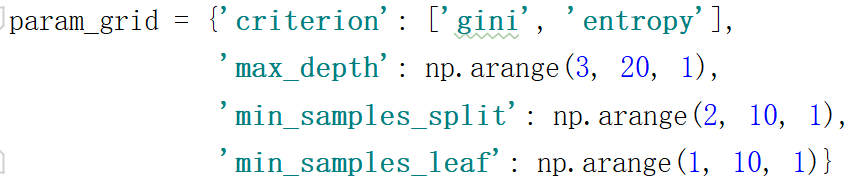
## Decision Trees

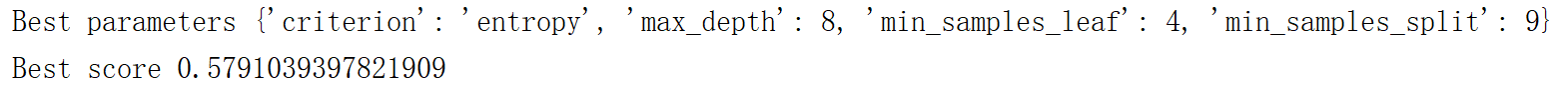
### Default model



This result shows the characteristic of decision trees: good at considering all possibilities but tend to overfit data. And there is no need to scale.

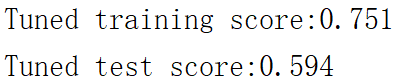
### Tune hyper-parameters





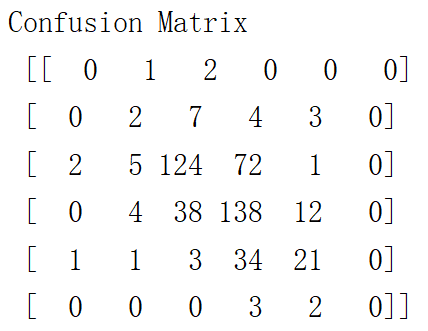
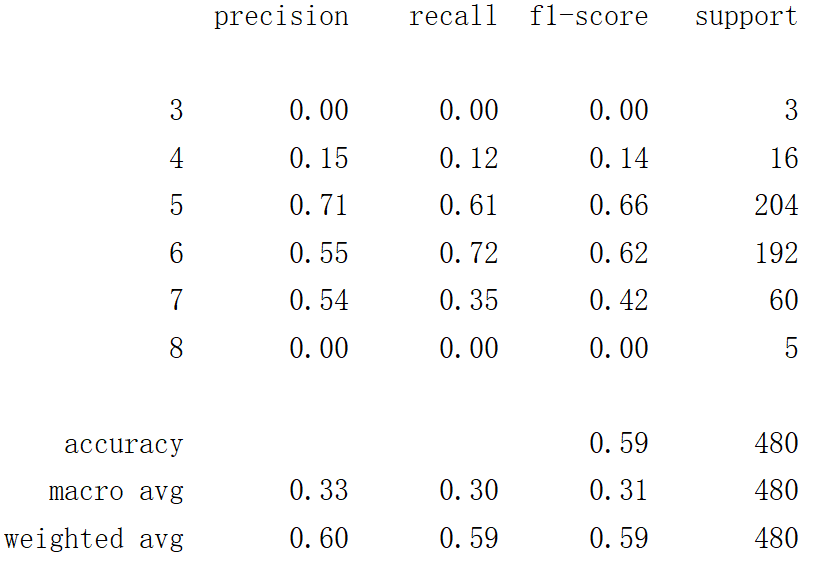
### Optimal model

1. Predict



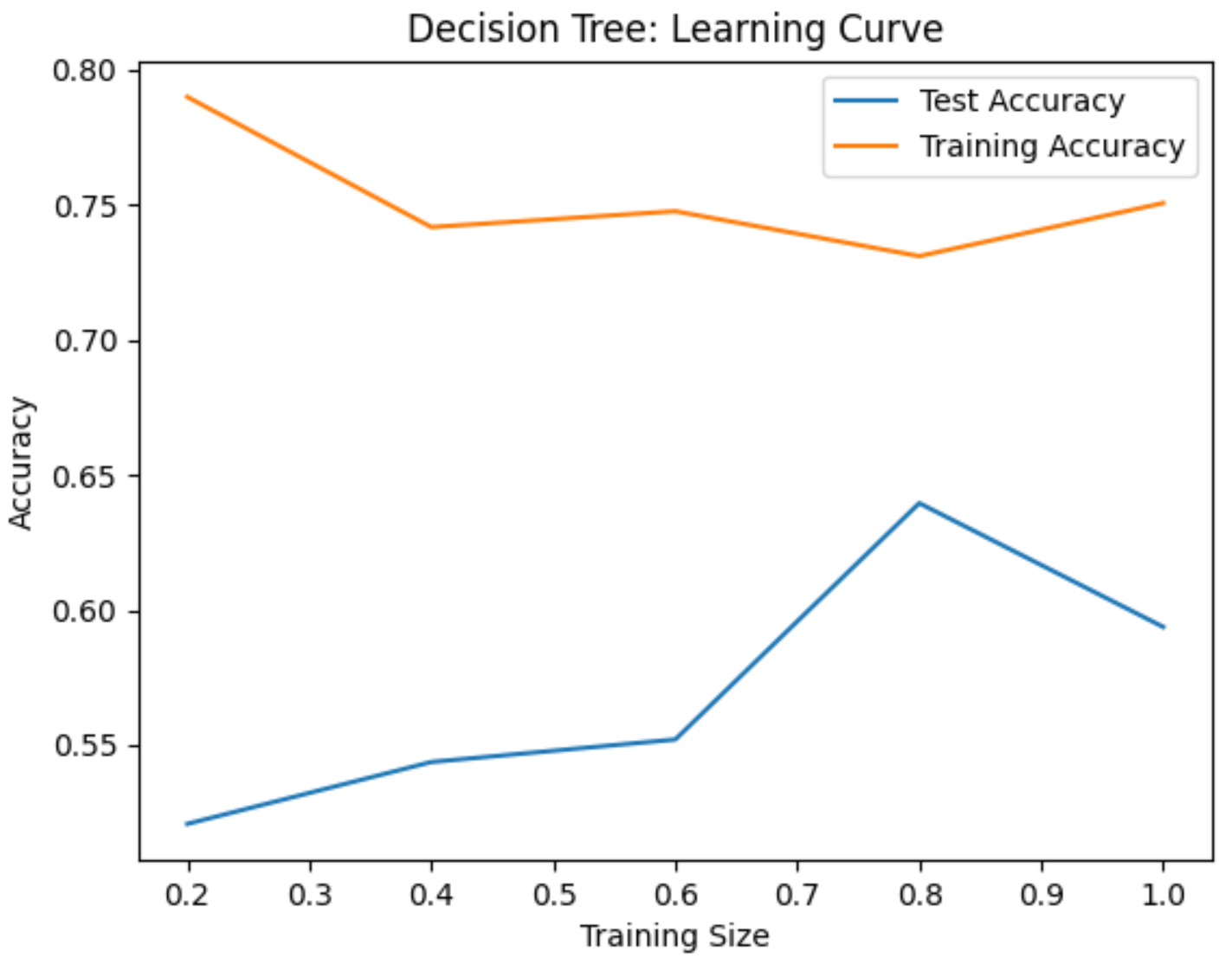
After pruning, overfitting has been alleviated a little.

1. Confusion matrix

See the score for class 4, the reason why Decision tree got a better result than KNN is that it can make predictions for small classes.

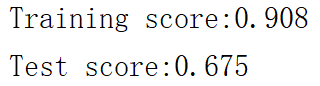
1. Learning curve



## Boosting

### Default model

I used Gradient boosting classifier:



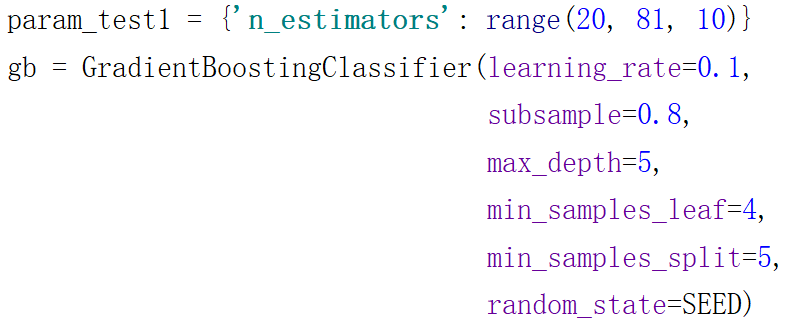
Since the weak learner Gradient boosting used is Decision tree. It has Decision tree’s advantage and disadvantage. By combining multiple weak learners, variance and bias will be reduced.

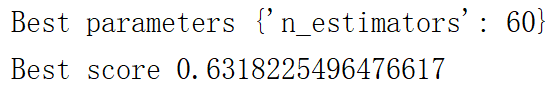
And there is no need to scale as well.

### Tune hyper-parameters

Procedures:

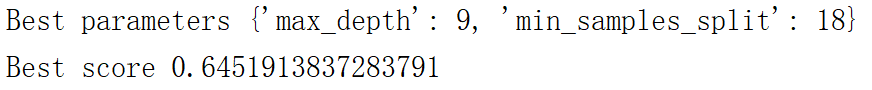
1. Fix learning\_rate and Decision tree parameters to search for optimal n\_estimators





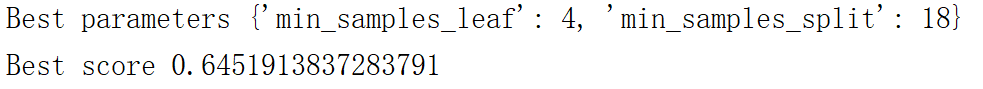
1. Fix learning\_rate and n\_estimators to search for optimal Decision tree parameters



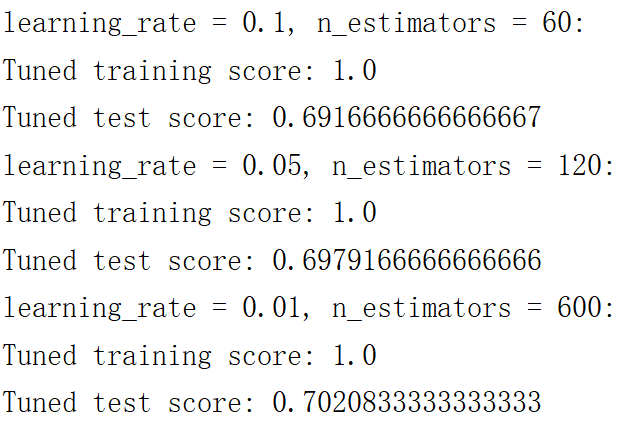


Fix max\_depth





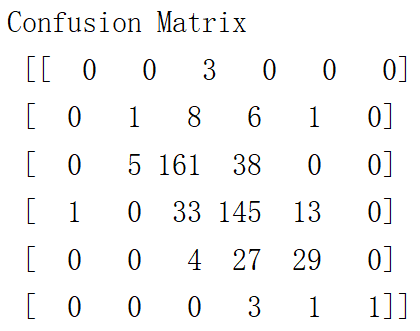
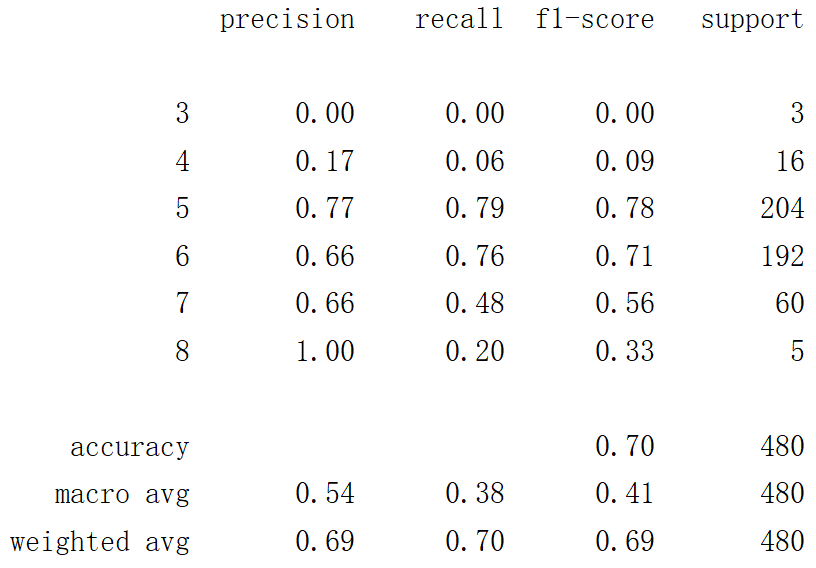
1. Lower learning\_rate and increase n\_estimators to make the model more robust



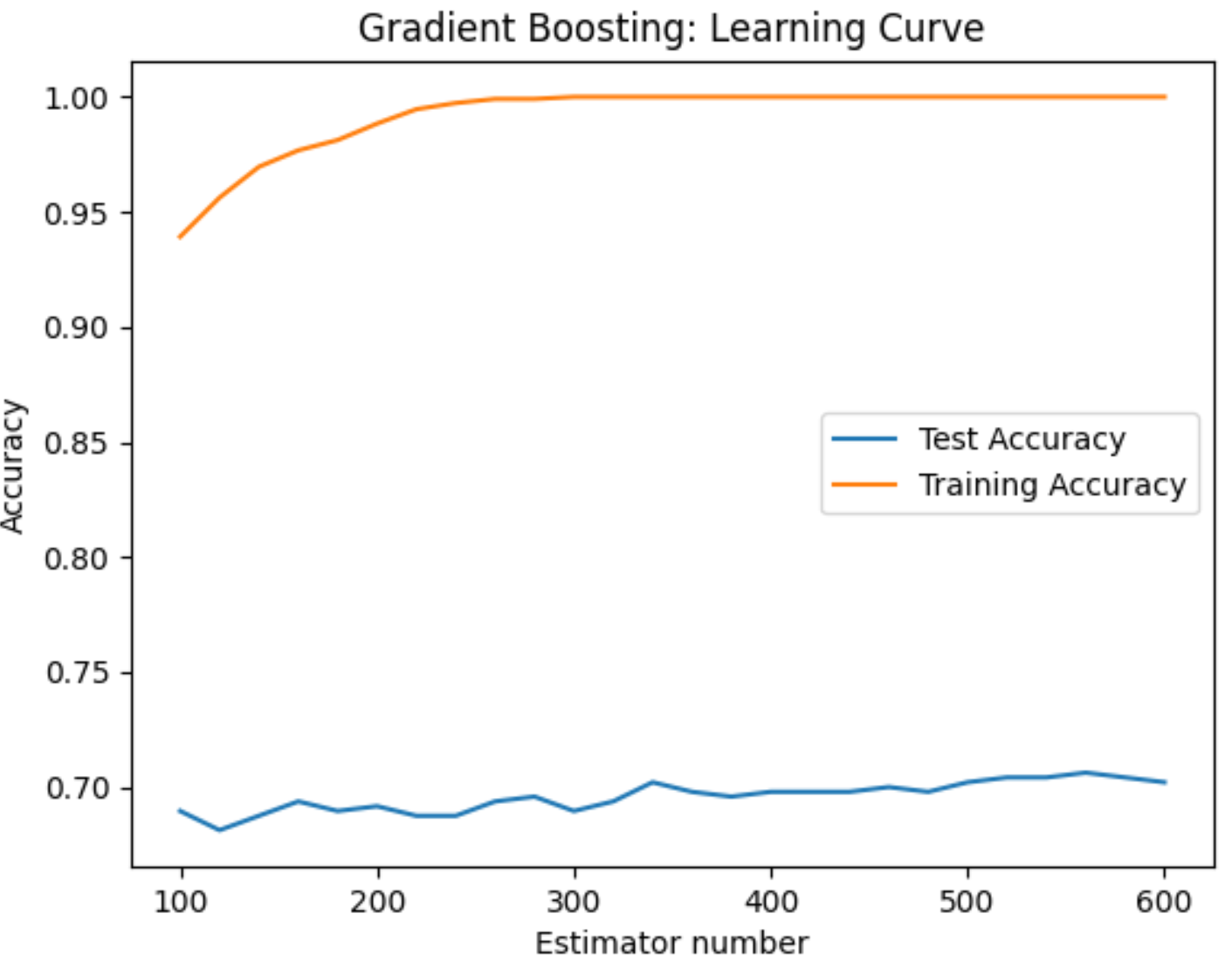
The result is significantly better than Decision tree. While as I used larger number of estimators, the training time increased significantly as well.

### Optimal model

1. Confusion matrix

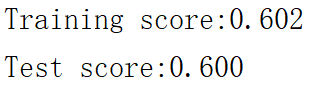
1. Learning curves

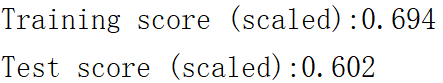


Just a little bit improvement for test accuracy.

## Neural Networks

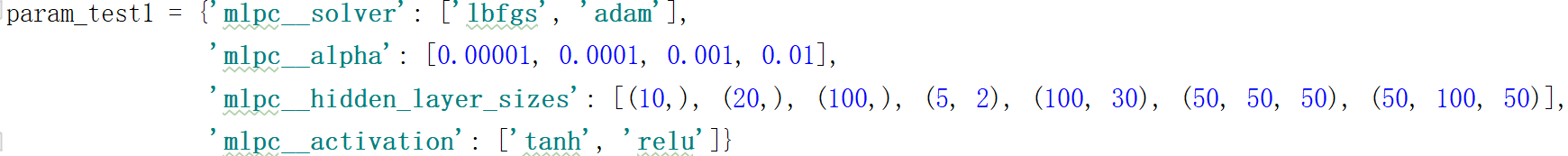
### Default model

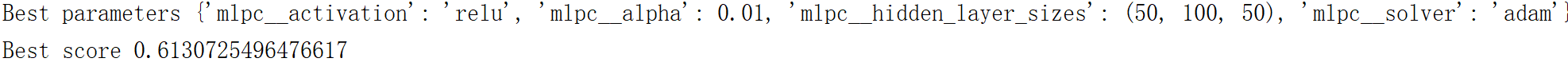




Since the scaled model has not converged completely, if I set a larger max\_iter, it will have an even better score. As a result, Neural networks will benefit from scaling.

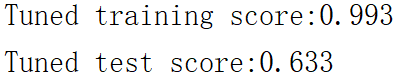
### Tune hyper-parameters



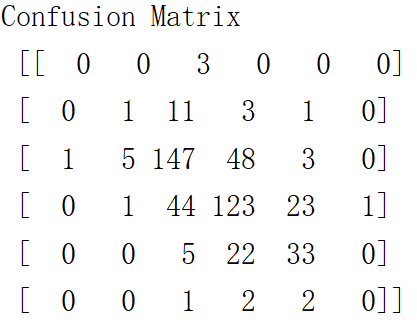
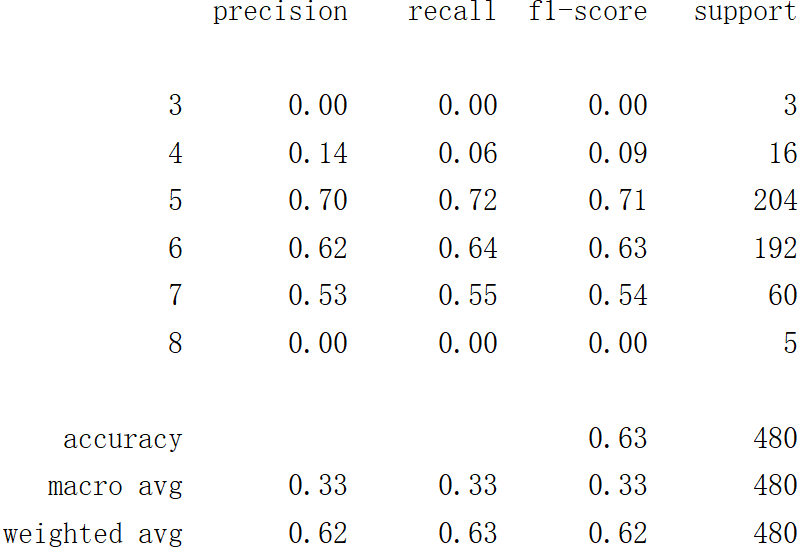


### Optimal model

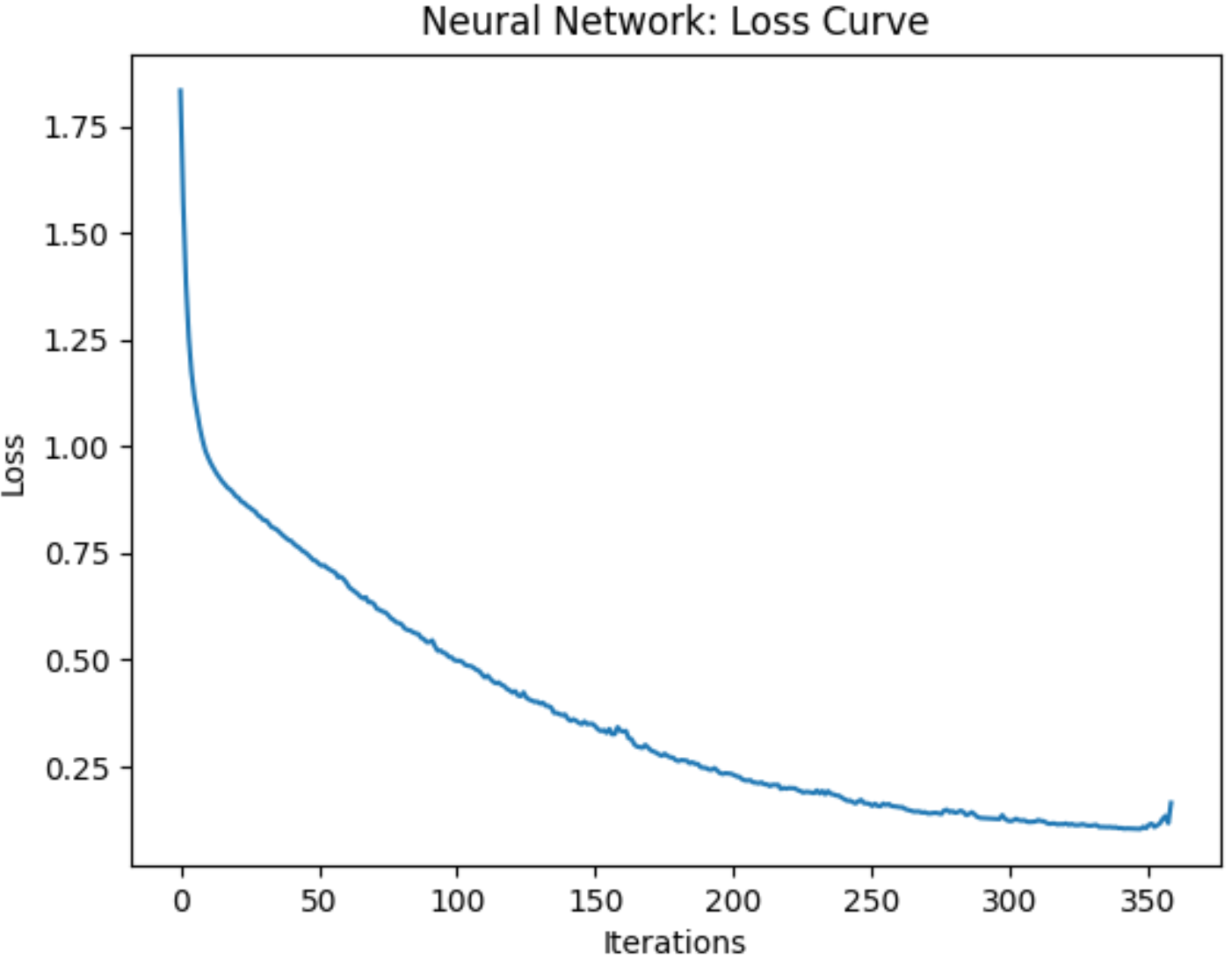
1. Predict



1. Confusion matrix

1. Learning curves

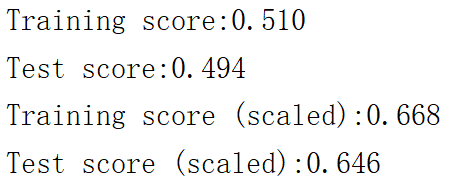


## Support Vector Machines

### Default model

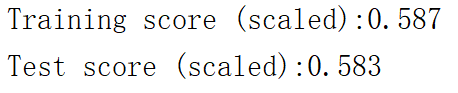
I tried three kernels:

1. Radial base function kernel



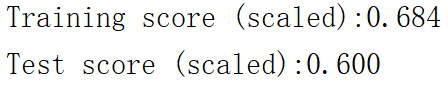
It’s clear that SVM need scaling.

1. Linear kernel



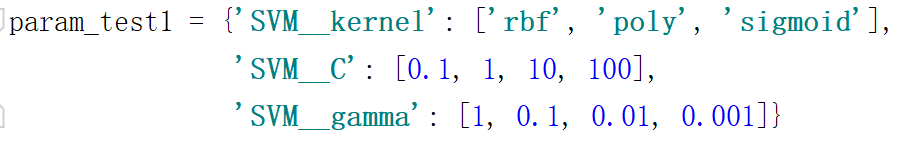
There is no hyper-parameter for linear kernel. This is the best it can get. So I skipped it.

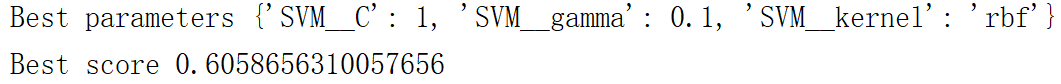
1. Polynomial kernel



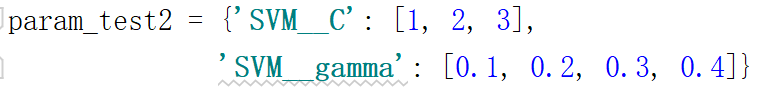
Just a little worse than RBF, I chose it as candidate.

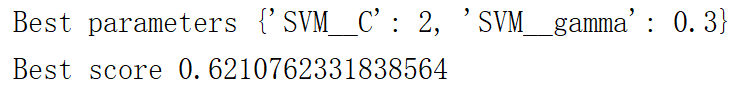
### Tune hyper-parameters





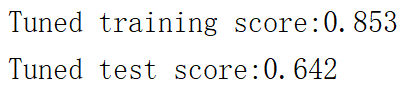
Choose RBF kernel:



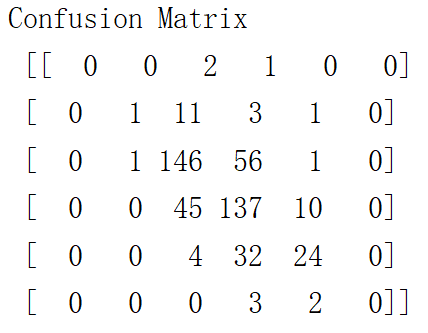
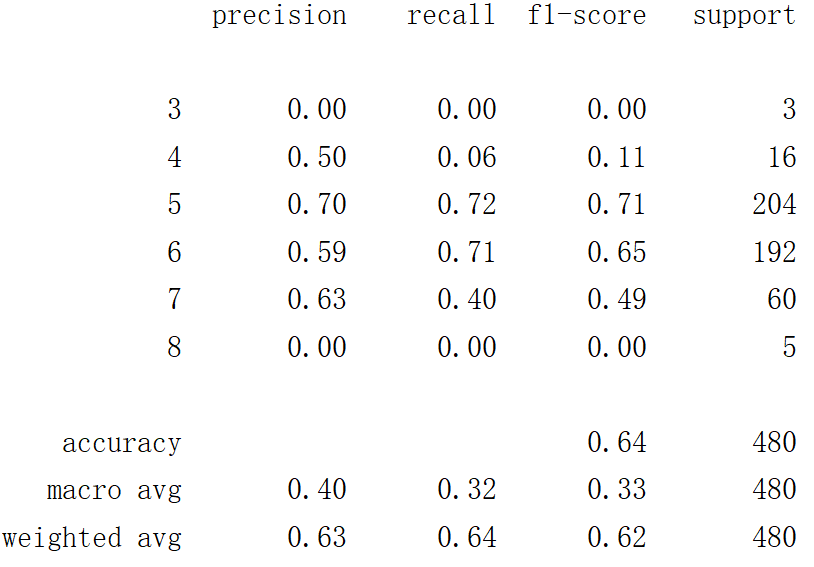


### Optimal model

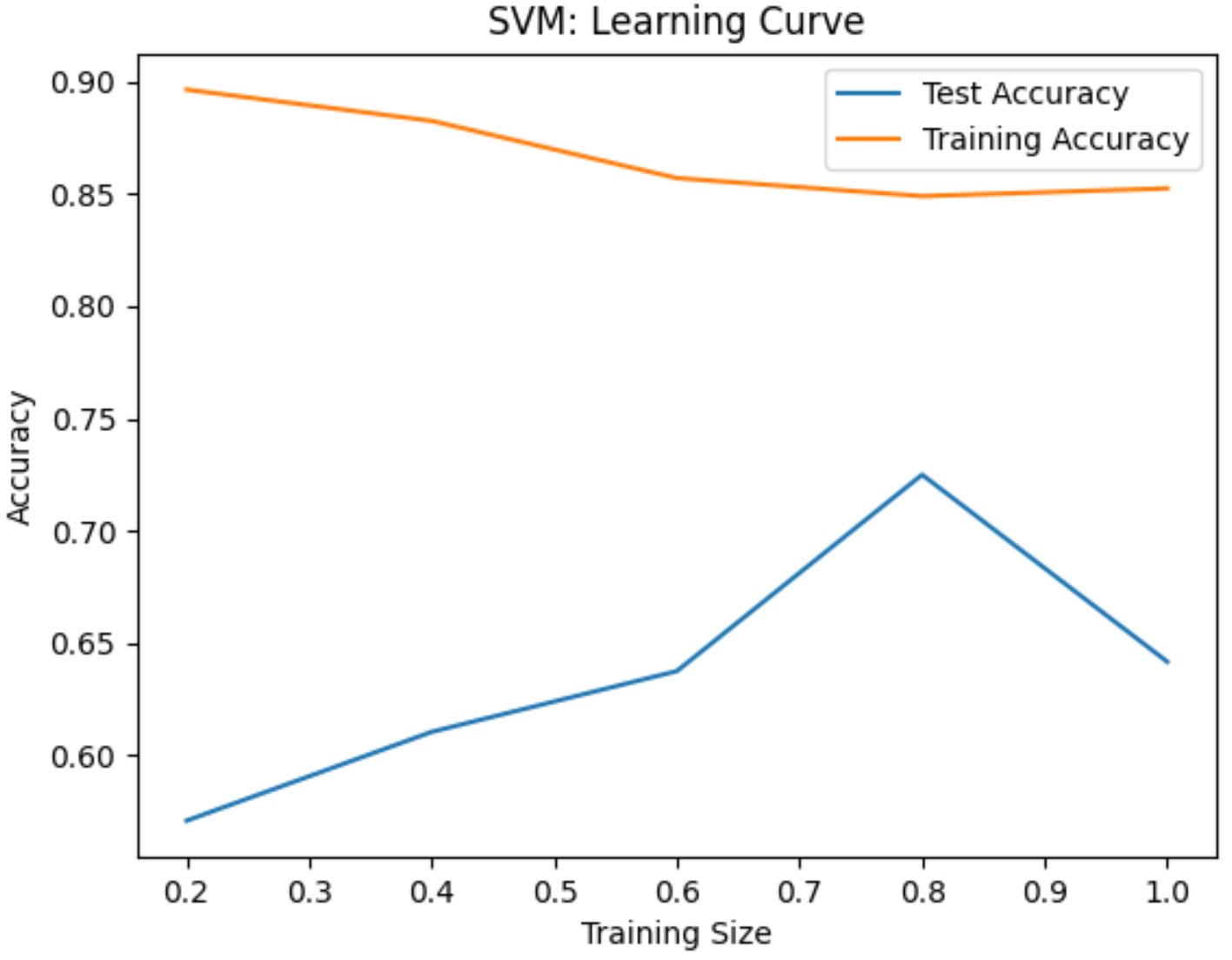
1. Predict



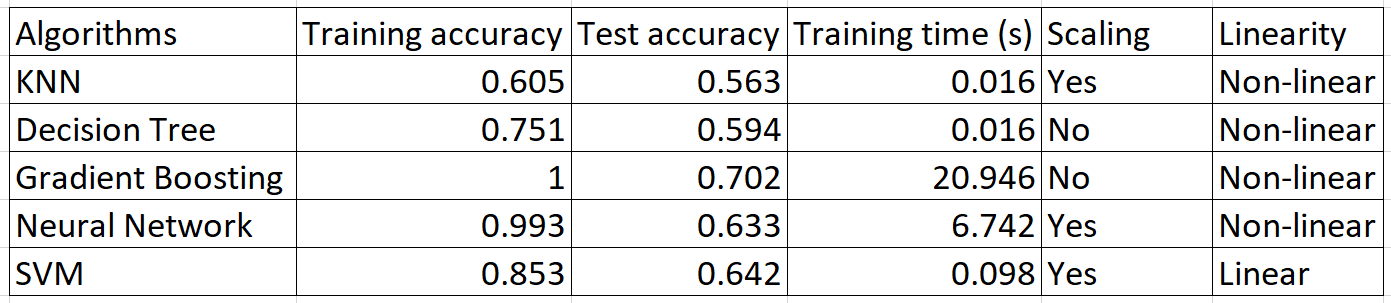
1. Confusion matrix

1. Learning curves



## In conclusion



Boosting got the best score, while since it uses a chain of estimators, the training time is the longest.

Advanced algorithms like Neural networks and SVM performed better. Because Neural networks use an iterative approach, the training time is longer than SVM.

KNN being affected the most by unbalanced classes.

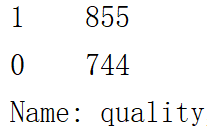
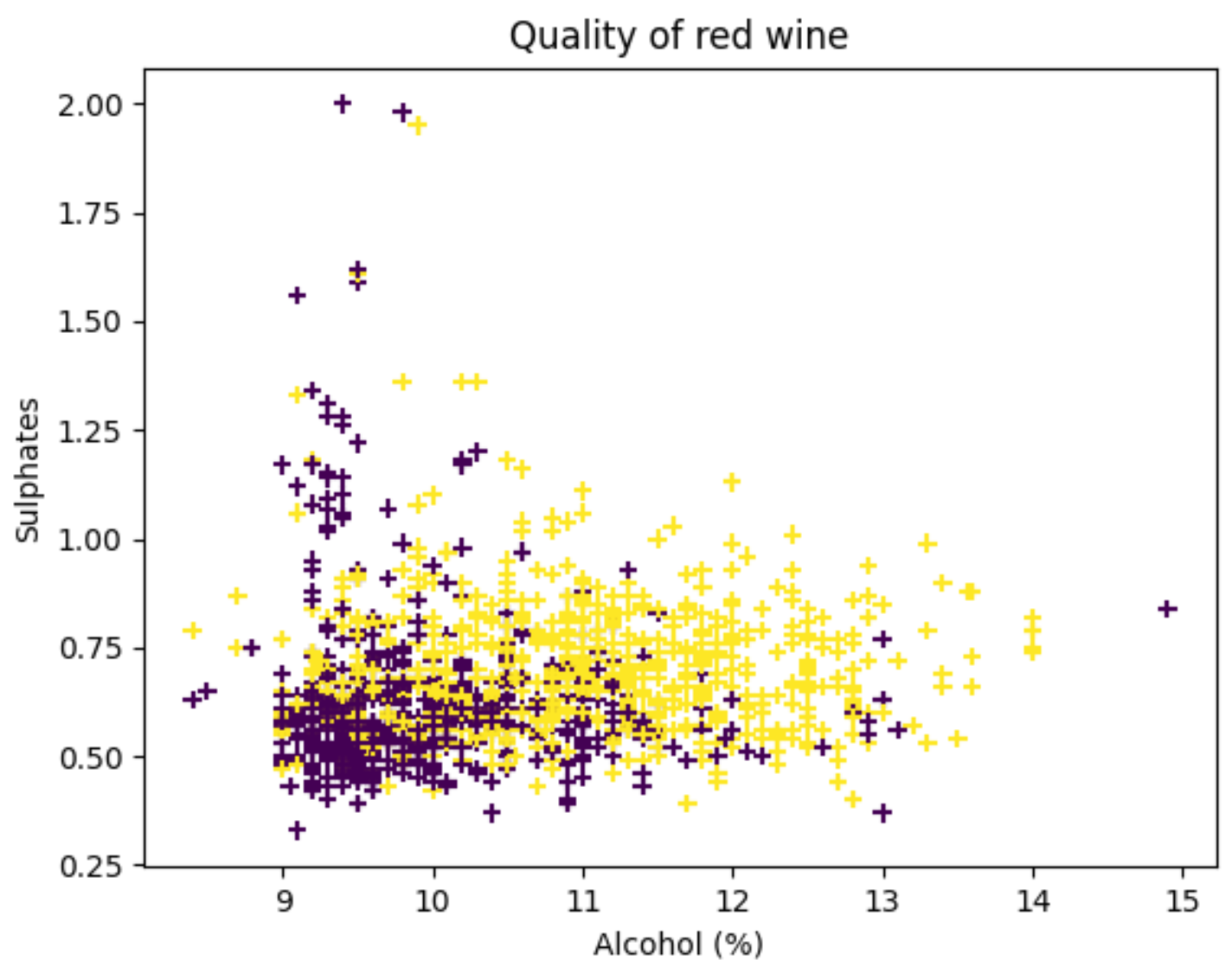
# Problem 2

## Problem statement

### Dataset

I used the same dataset as Problem 1. Instead, I turned the 10 point scale into dichotome variable: quality that 6 or higher as good/1, and the rest as not good/0.

### Visualize

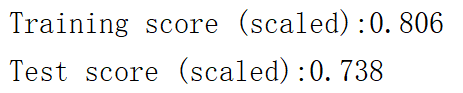
 

### Split training / test set

I split the dataset using the same strategy as in Problem 1.

## k-Nearest Neighbors

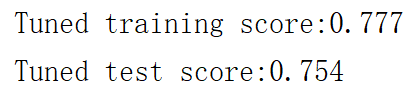
### Default model



Much better than the results of multi-classification problem.

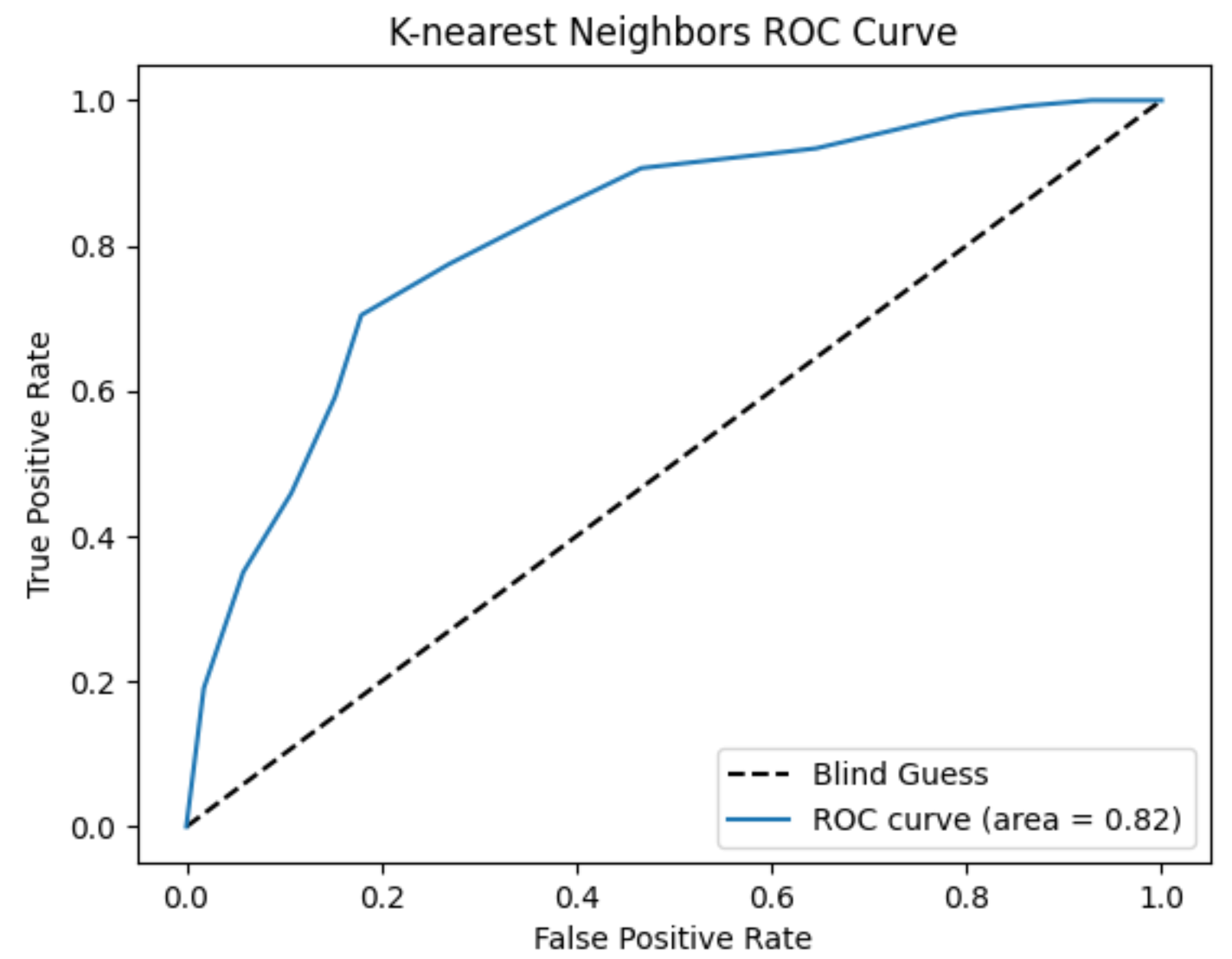
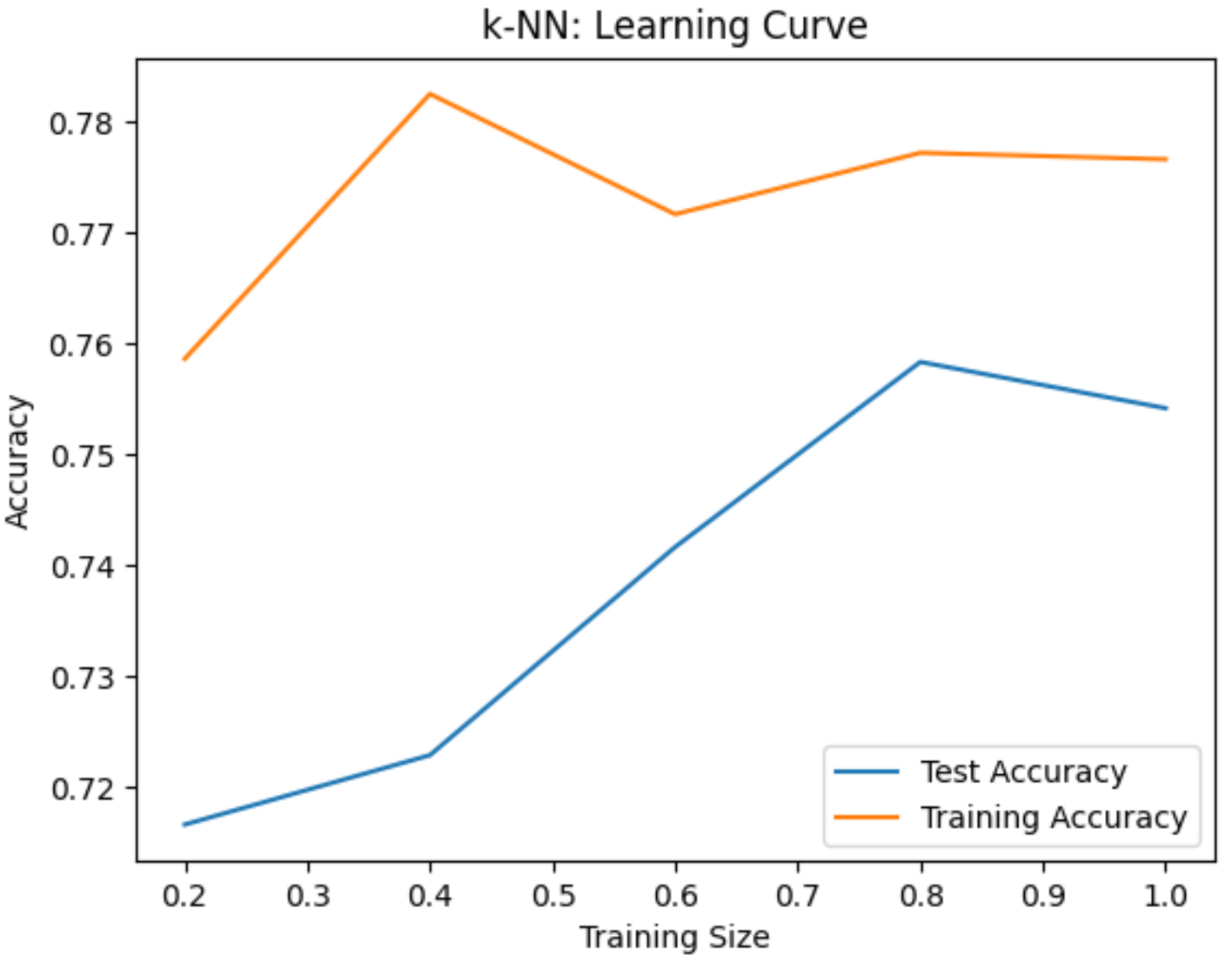
### Optimal model

1. Predict



Although training score decreased with a larger K - 12 used, the ability to generalize increased. Compared to Problem 1, KNN performed much better when the classes are balanced.

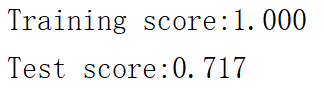
1. ROC curve and learning curve

We now can see the trend that the model learns better with larger training set.

## Decision Trees

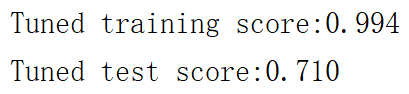
### Default model



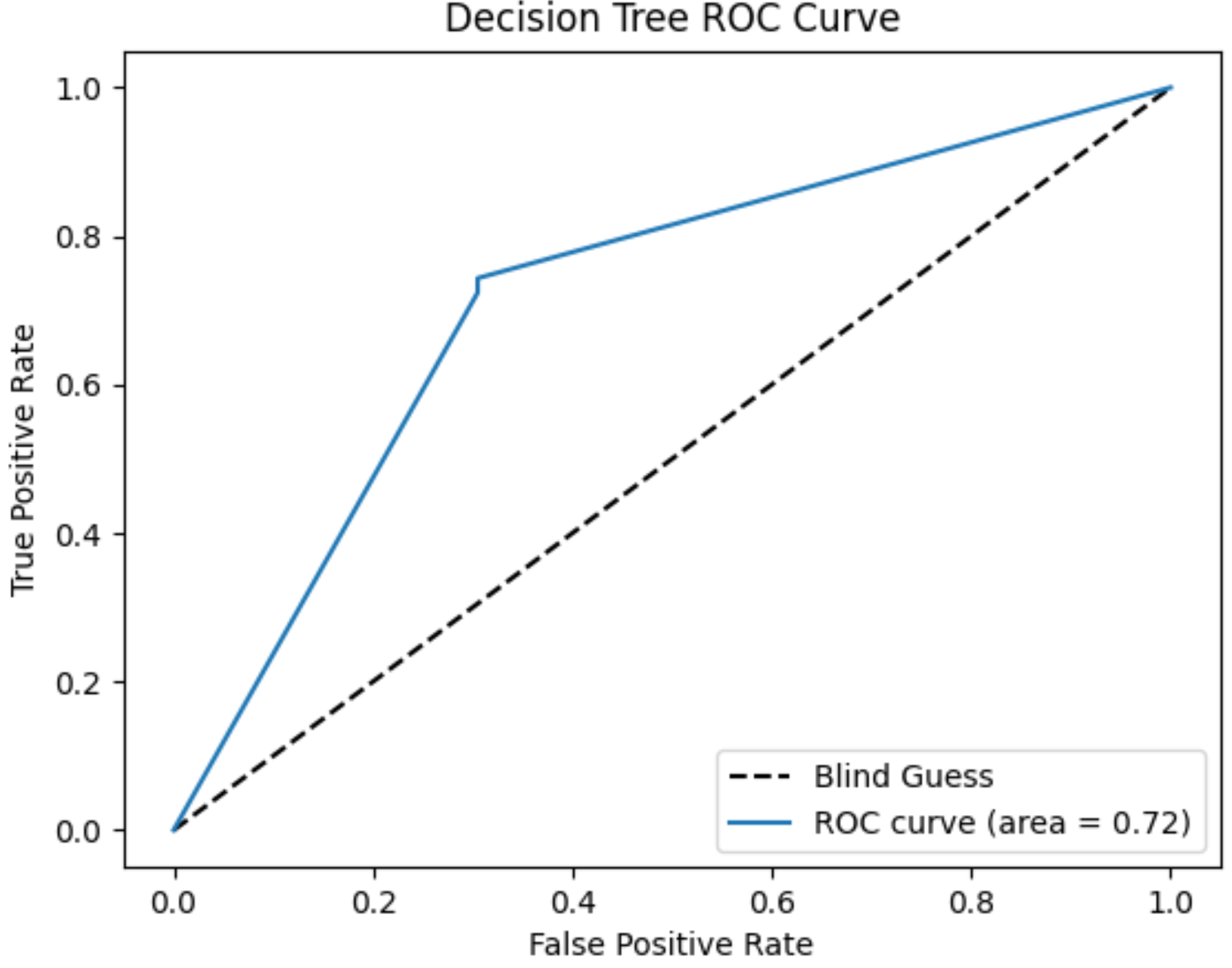
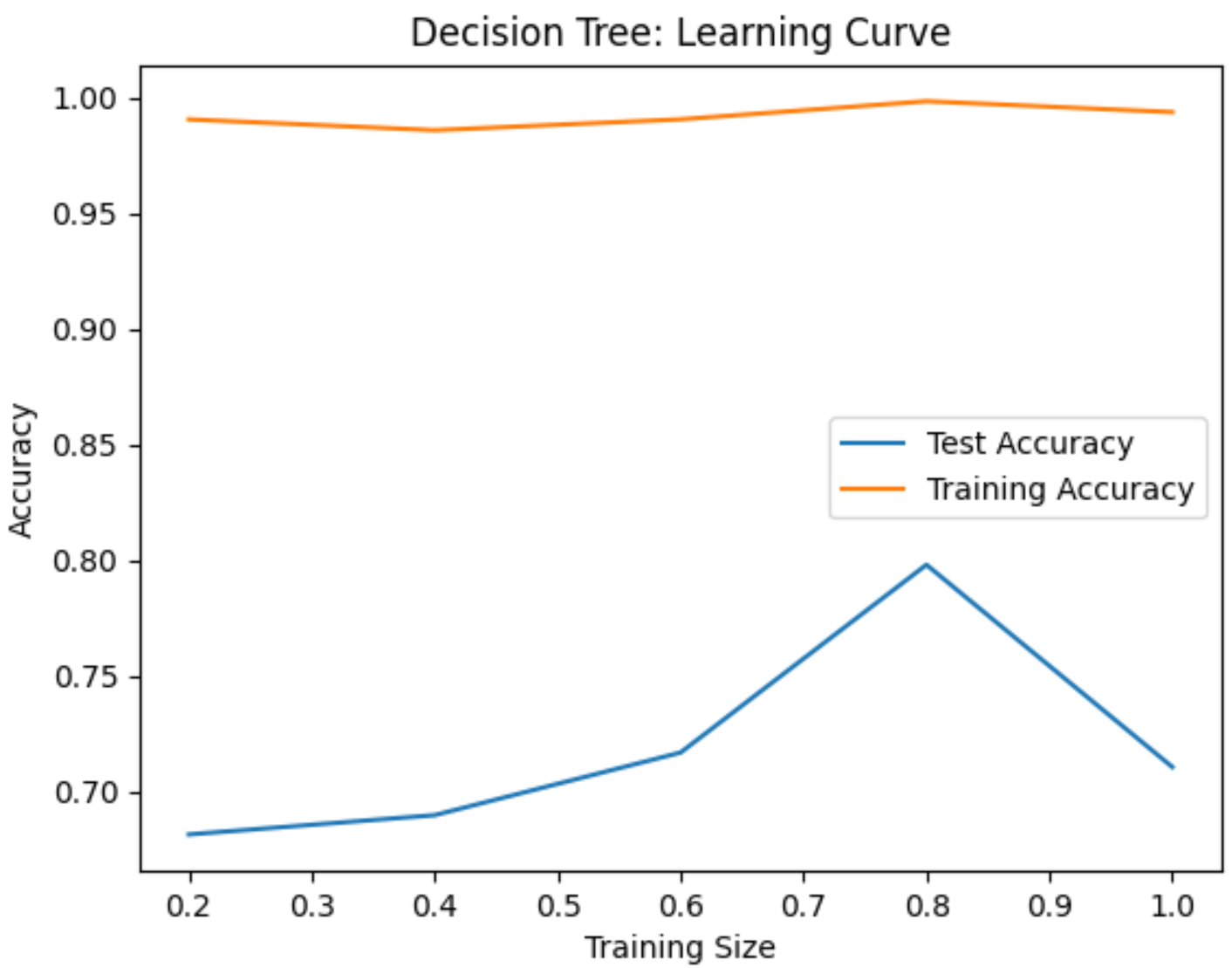
Overfitting as usual if we don’t prune it.

### Optimal model

1. Predict



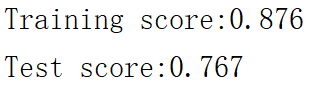
1. ROC curve and learning curve

## Boosting

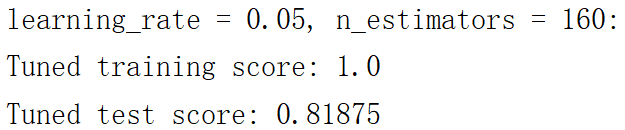
### Default model

I used Gradient boosting as well:

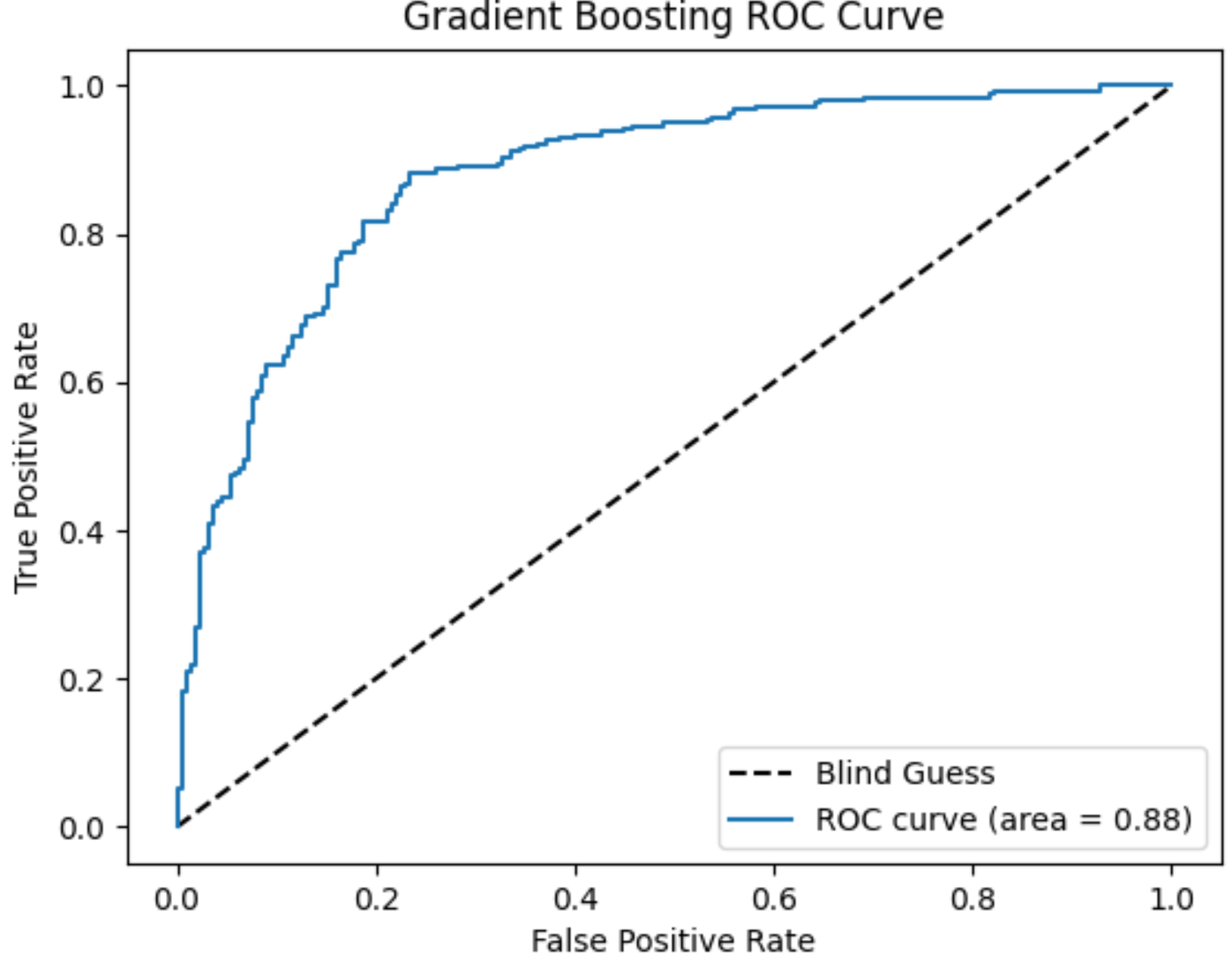
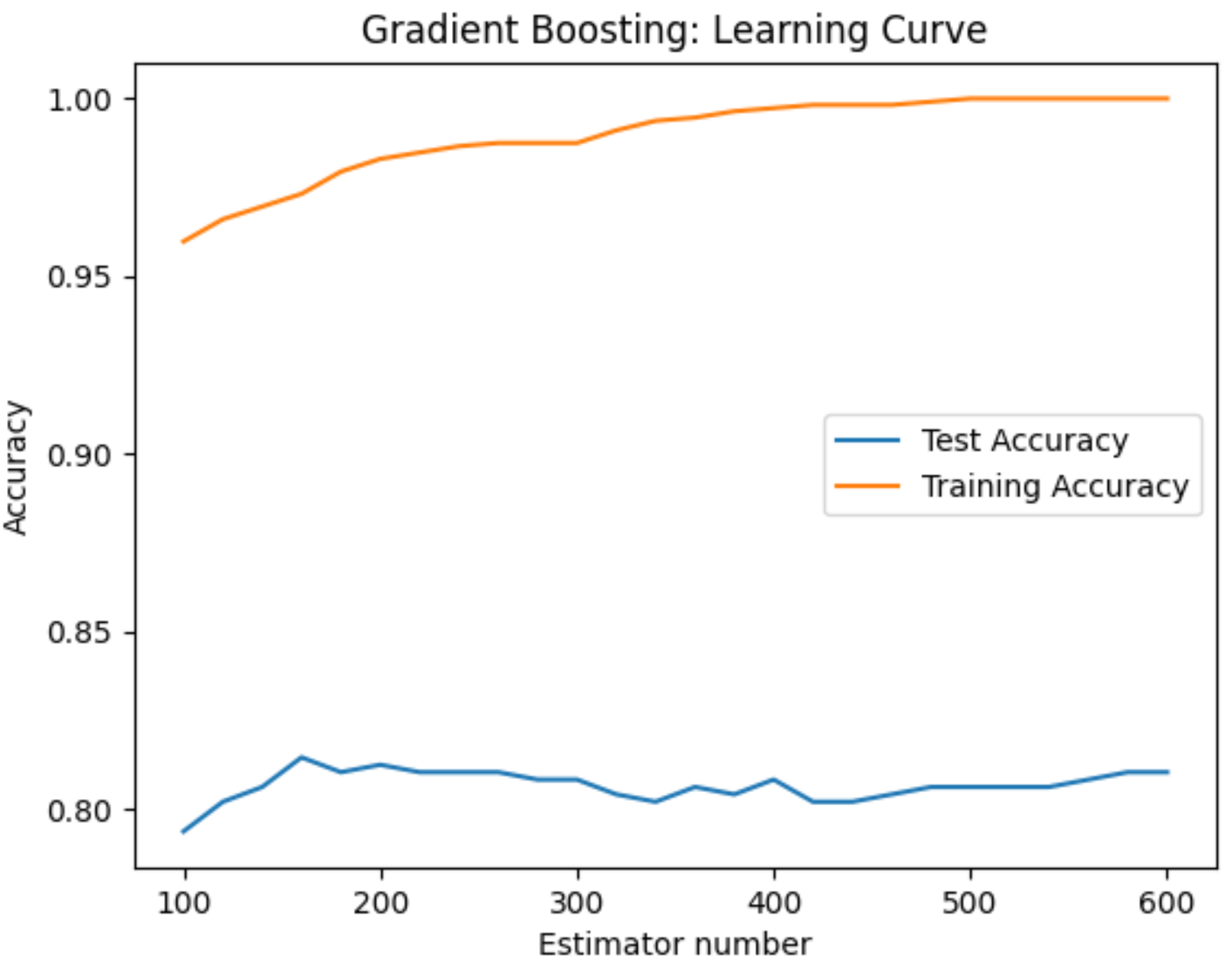


### Optimal model

1. Predict

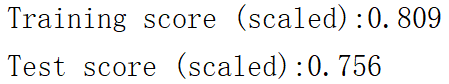


1. ROC curve and learning curve

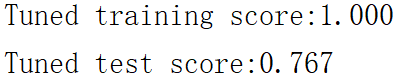
## Neural Networks

### Default model

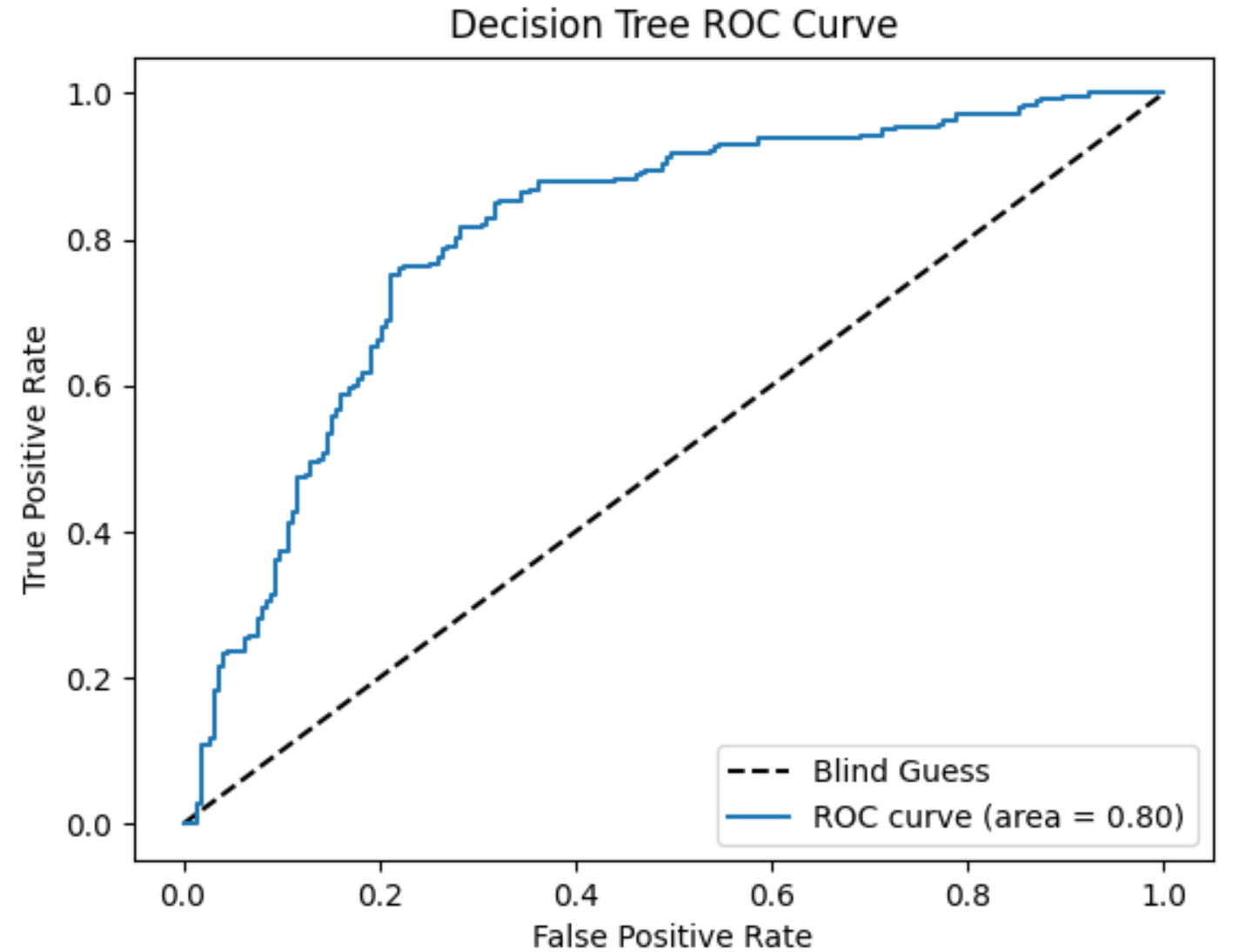
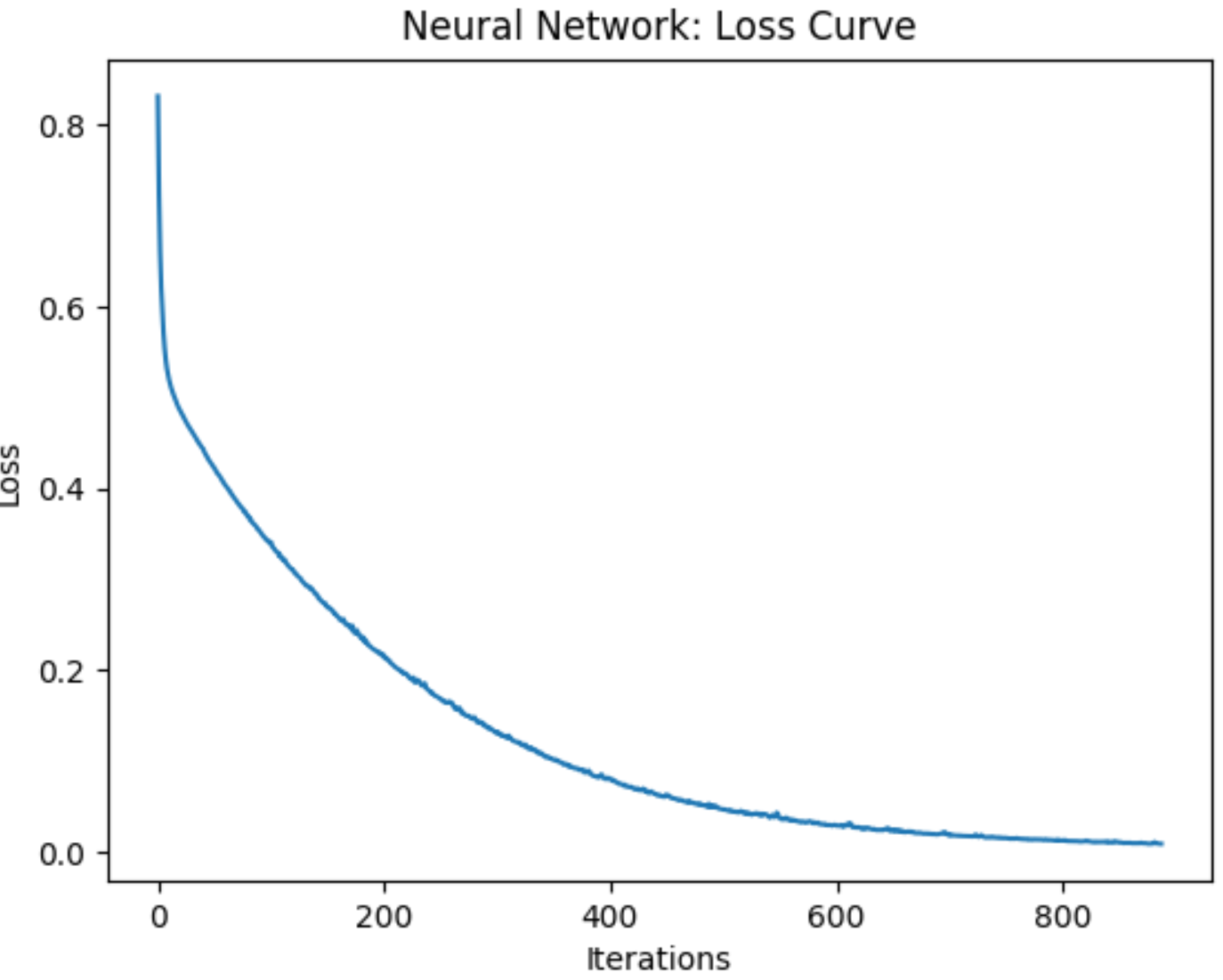


### Optimal model

1. Predict



1. ROC curve and learning curve

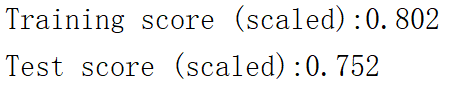
 

## Support Vector Machines

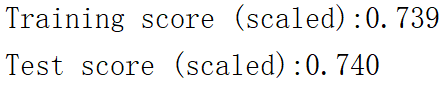
### Default model

I tried three kernel as well:

1. Radial base function kernel

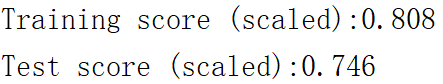


1. Linear kernel



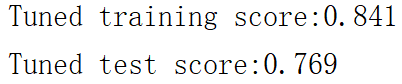
It seems that a two-dimensional linear model is not suitable for this problem.

1. Polynomial kernel

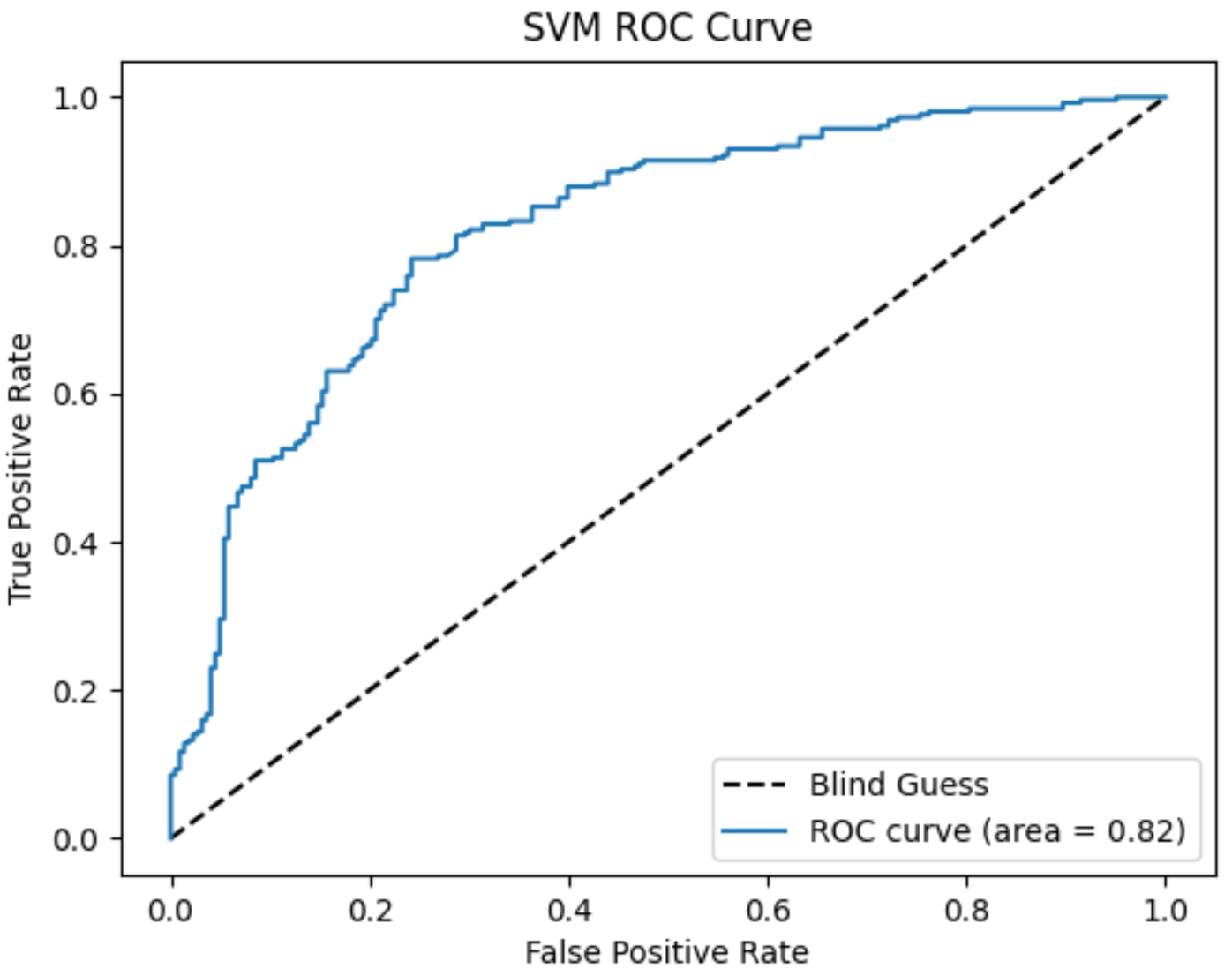
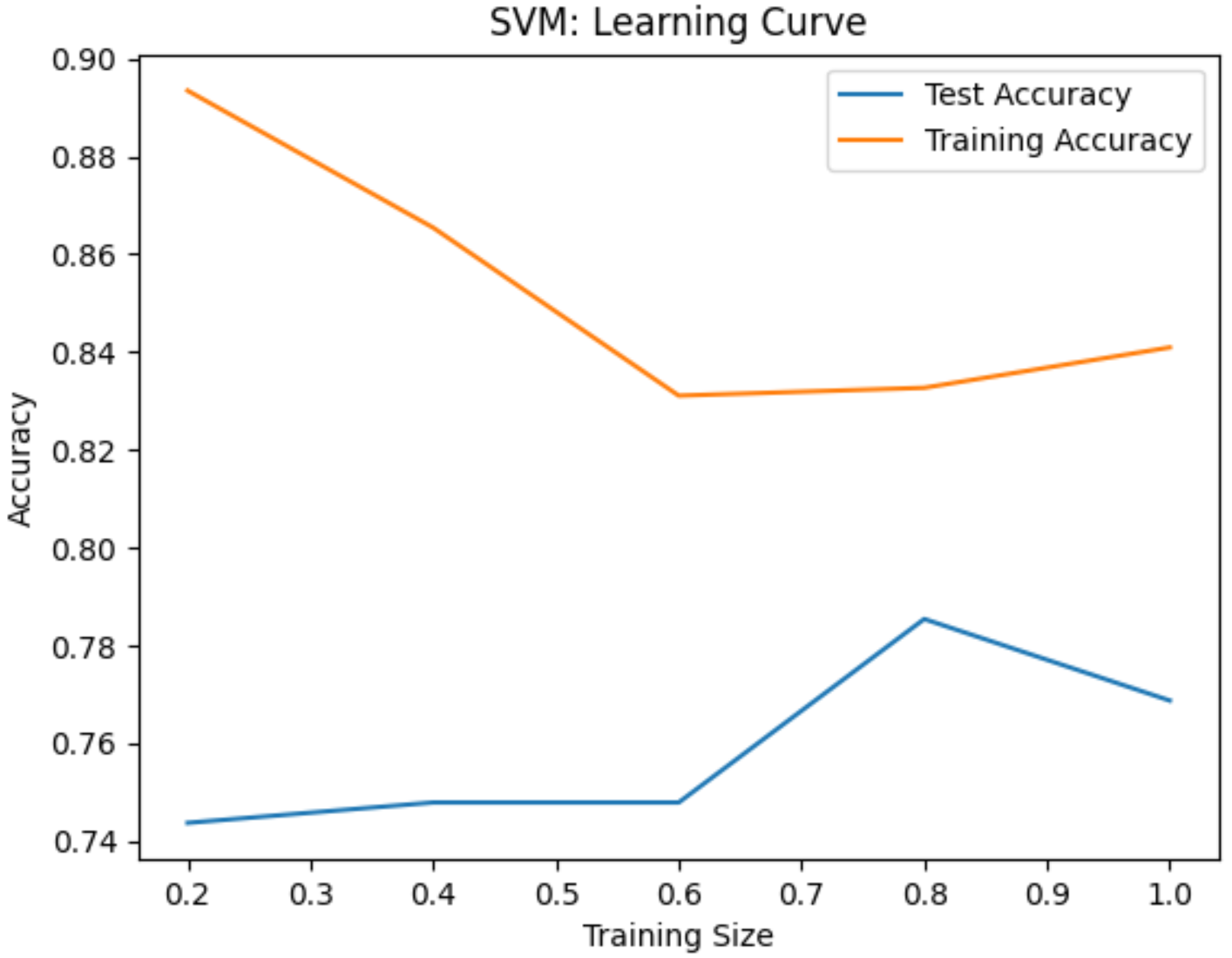


### Optimal model

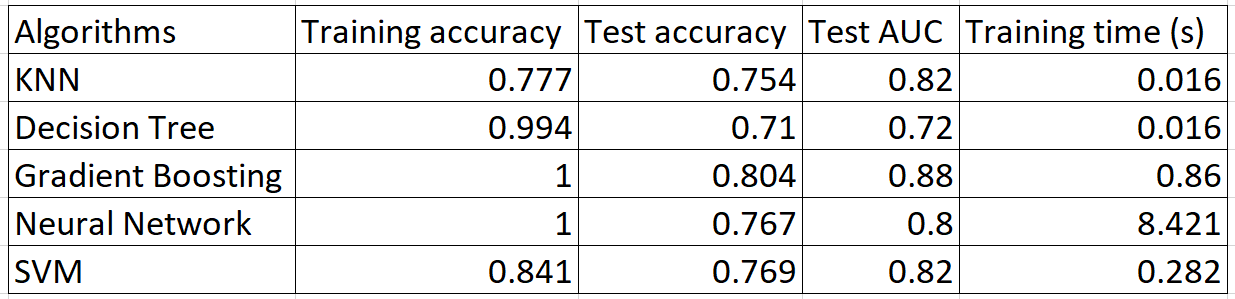
1. Predict



1. ROC curve and learning curve

## In conclusion



Compared to the results of Problem 1, unbalanced classes indeed affect the accuracy. However, scores of Problem 2 are still not good enough. It might because there are only physicochemical features are available in the dataset. We probably need more relevant features like grape types, wine brand, etc.

# Reference

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties.   
In Decision Support Systems, Elsevier, 47(4):547-553, 2009.