Comparison of Supervised Learning Models:

Case Studies with Two Classification Problems

Jenny Hung

(Jhung34)

Table of Contents

[Introduction 2](#_Toc493687324)

[Two Classification Problems Considered 2](#_Toc493687325)

[Freddie Mac mortgages loan-level performance data 2](#_Toc493687326)

[Blood donation prediction 3](#_Toc493687327)

[Training and Test Results 4](#_Toc493687328)

[Freddie Mac loan-level dataset 4](#_Toc493687329)

[Learning curve analysis 4](#_Toc493687330)

[Complexity analysis 5](#_Toc493687331)

[Blood donation prediction dataset 6](#_Toc493687332)

[Learning curve analysis 6](#_Toc493687333)

[Complexity analysis 7](#_Toc493687334)

[Analysis of models 8](#_Toc493687335)

[Freddie Mac loan-level dataset 8](#_Toc493687336)

[Blood donation prediction dataset 9](#_Toc493687337)

# Introduction

We implemented the following five learning algorithms:

* Decision tree with pre-pruning
* Neural network
* Gradient Boosting
* Support Vector Machine
* K Nearest Neighbors

In addition, we introduced two separate classification problems as separate platforms to deploy the learning algorithms. Training and testing errors using each of the learning algorithms are then recorded and analyzed.

We showed that there is no free lunch in data science, in that the classifier that provides the best performance for one problems does not necessarily deliver the same level of performance for the other problem.

# Two Classification Problems Considered

## Freddie Mac mortgages loan-level performance data

Freddie Mac, one of the government-sponsored enterprises went through a nearly [$200 billion government bailout](http://en.wikipedia.org/wiki/Federal_takeover_of_Fannie_Mae_and_Freddie_Mac) during the financial crisis, motivated in large part by losses on loans that they guaranteed, so I figured there must be something interesting in the loan-level data. Freddie Mac began reporting loan-level credit performance data in 2013 at the direction of their regulator, the Federal Housing Finance Agency. The stated purpose of releasing the data was to “increase transparency, which helps investors build more accurate credit performance models in support of potential risk-sharing initiatives.”

I decided to dig into this dataset – to analyze multiple-period mortgage risk at loan level using the Freddie Mac dataset prime and subprime mortgages originated in the United States in 2016, which includes the individual characteristics of each loan, and monthly updates on loan performances over life of a loan.

The entire Freddie Mac dataset is immensely rich, in that it encompass a 10-year span and contains millions of loan-level mortgage performance and default information. In this implementation, we have also considered a random subset of the loan-level, consists of mortgage performance and defaults recorded in 2016. The dataset in this implementation has more than 200,000 mortgages performance records.

This problem is interesting because there exists a highly nonlinear relationship between the variables and the default prediction. In particular, many of the existing academic research studied deep neural networks, which have multiple layers of hidden nodes. In this implementation, I am more interested in finding out the relative strengths, time complexity, sample complexity and mistake bounds associated with using each one of the classifiers in question.

## Blood donation prediction

The [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/index.html) is a great repository of data science-related projects. Given my interest in discovery our mission, we're interested in predicting if a blood donor will donate within a given time window. This is considered a beginnier’s dataset.

# Analysis of Models: Training and Test Errors

In general, a learning curve demonstrates how accuracy of a learning algorithm changes as a function of training size. The rationale is as follows: given a train/test set partition which gives a given sample size , we randomly select data points from the training set, apply the learning algorithm to this set, and evaluate the model accuracy on the test set to determine the accuracy. This accuracy is then stored for each given train/test set partition, then a graph is plotted. In my implementation, I adopted the learning\_curve() function from the scikit-learn package, which determines cross-validated training and test scores for different training set sizes. In particular, a cross-validation generator splits the whole dataset k times in training and test data. Subsets of the training set with varying sizes will be used to train the estimator and a score for each training subset size and the test set will be computed. Afterwards, the scores will be averaged over all k runs for each training subset size.

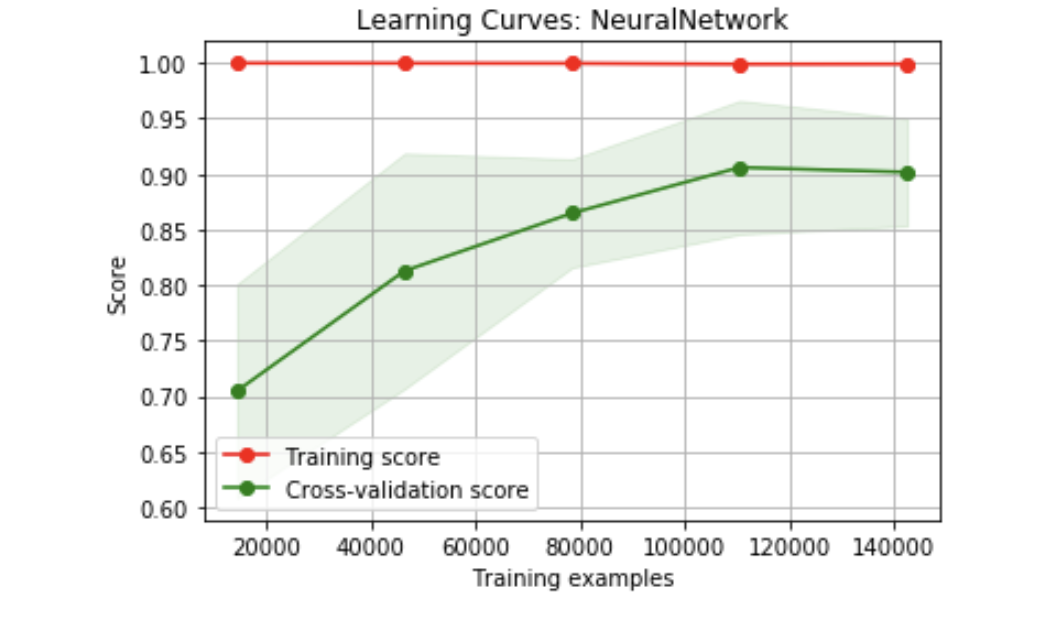
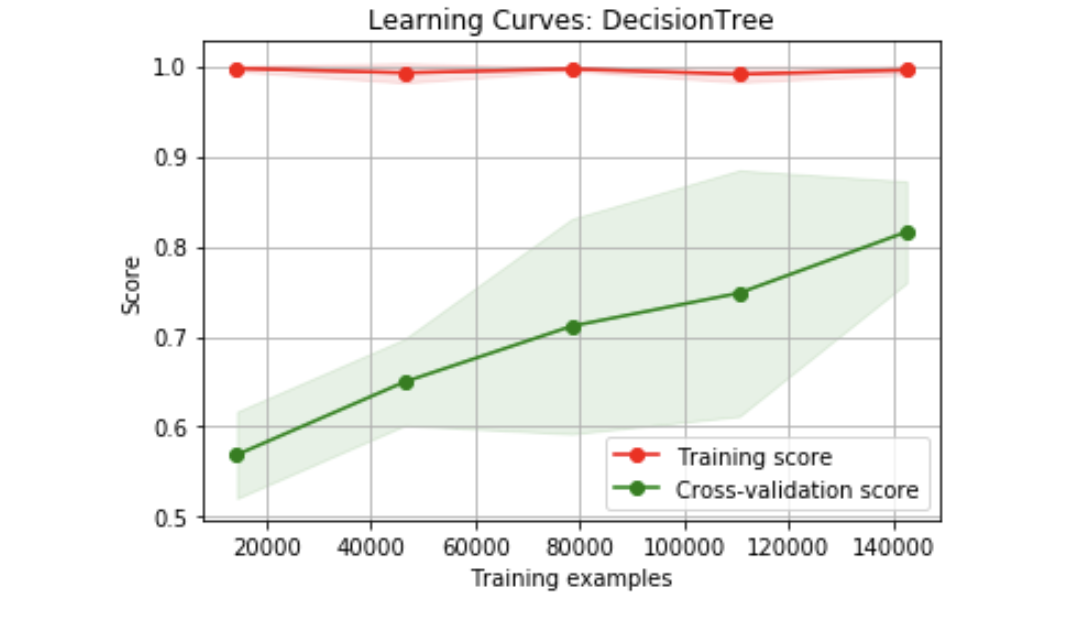
On the other hand, a complexity curve demonstrates how accuracy of a learning algorithm changes as a result of the tuning of hyper-parameters. In my implementation, I adopted the validation\_curve() method from the scikit-learn package, which determines training and test scores for varying parameter values, and compute scores for an estimator with different values of a specified parameter. This is similar to grid search with one parameter. However, my implementation will also compute training scores, in addition to being a plotting utility.

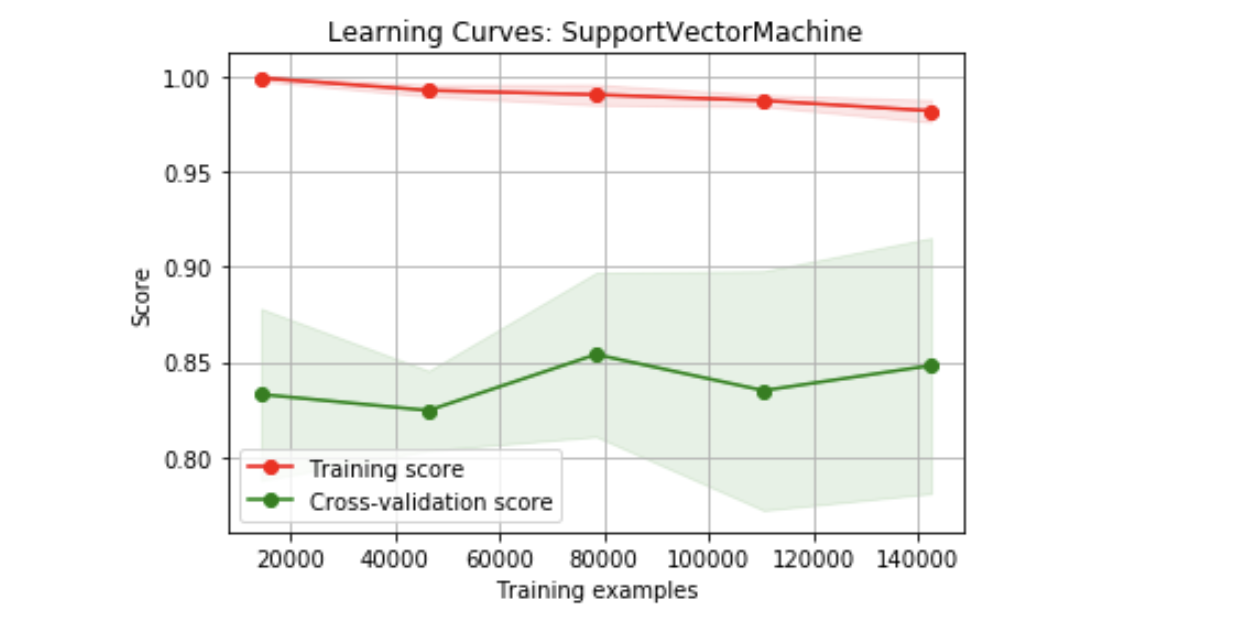
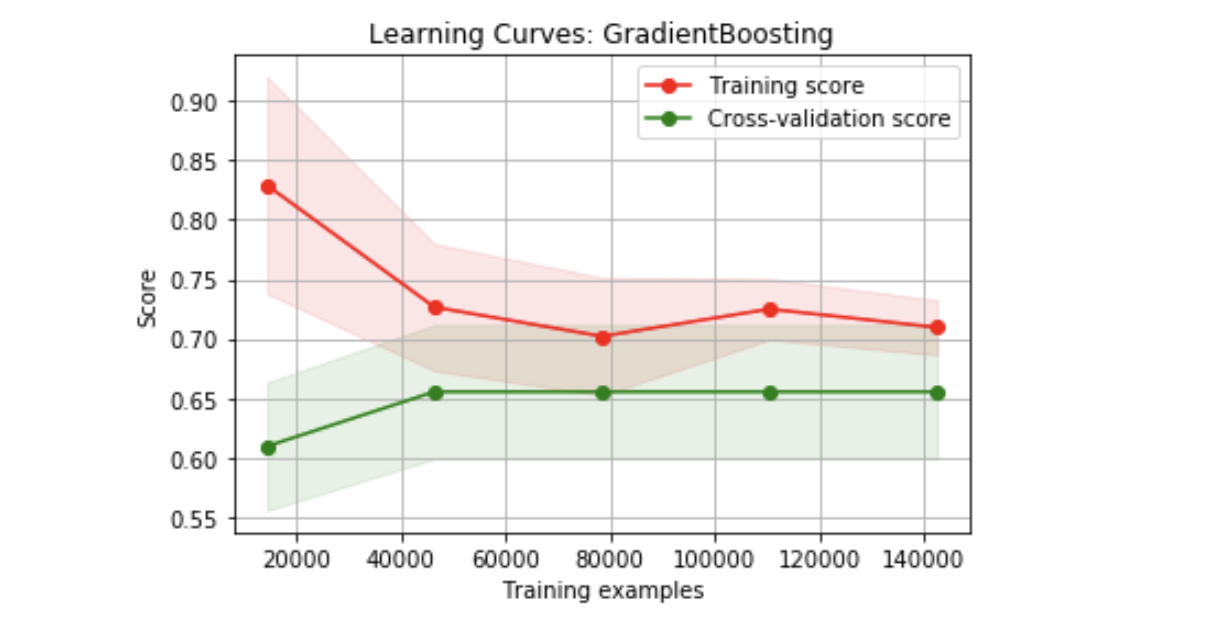
## Freddie Mac loan-level dataset

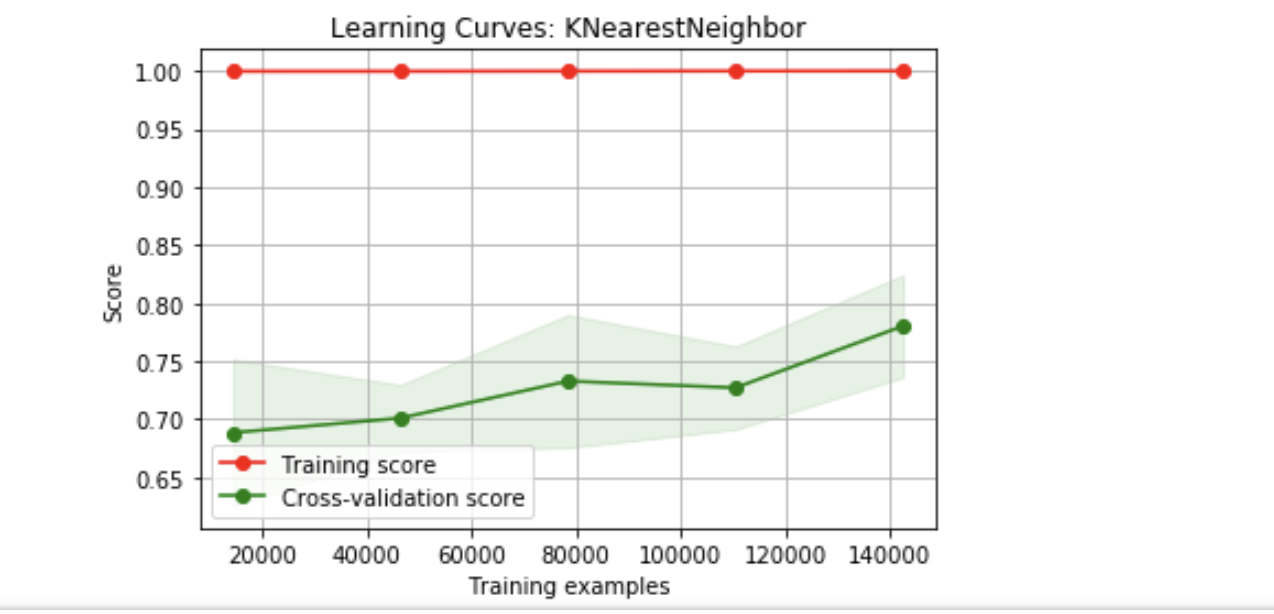
A significant challenge of the Freddie Mac dataset is that it is severely imbalanced, as it is much more commonplace that a mortgage loan is serviced on time as opposed to be in default. In my implementation, I had balanced the positive and negative classes whenever possible. Without balancing, the skew between positive and negative classes might have been so severe as to pose a challenge to any learning algorithm.

The graph is listed below:

### Learning curve analysis



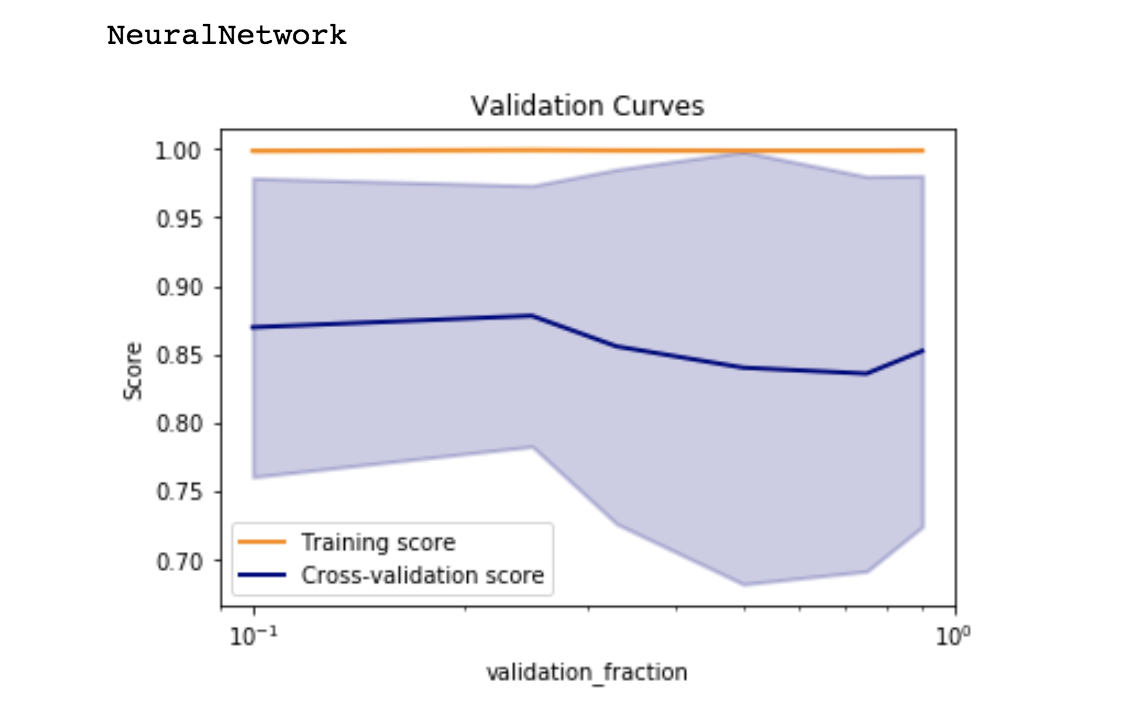
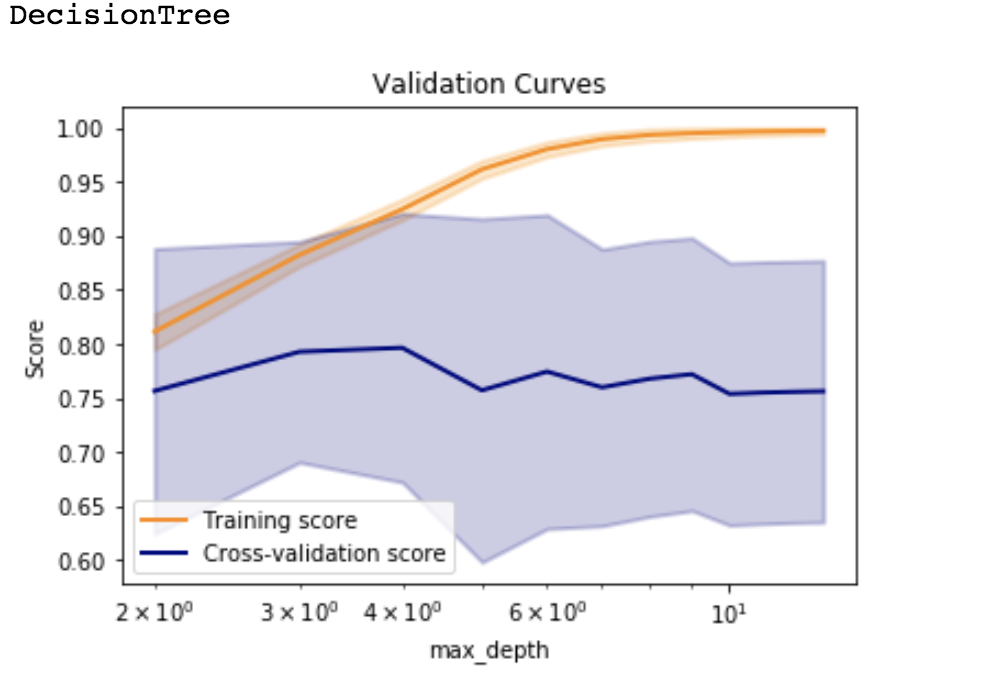


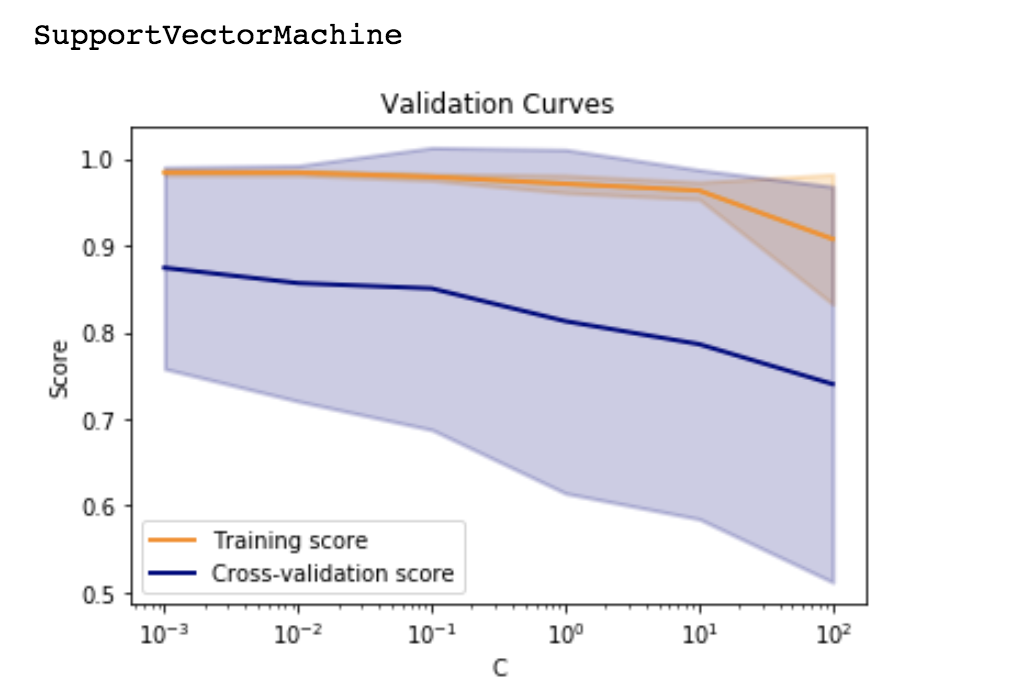
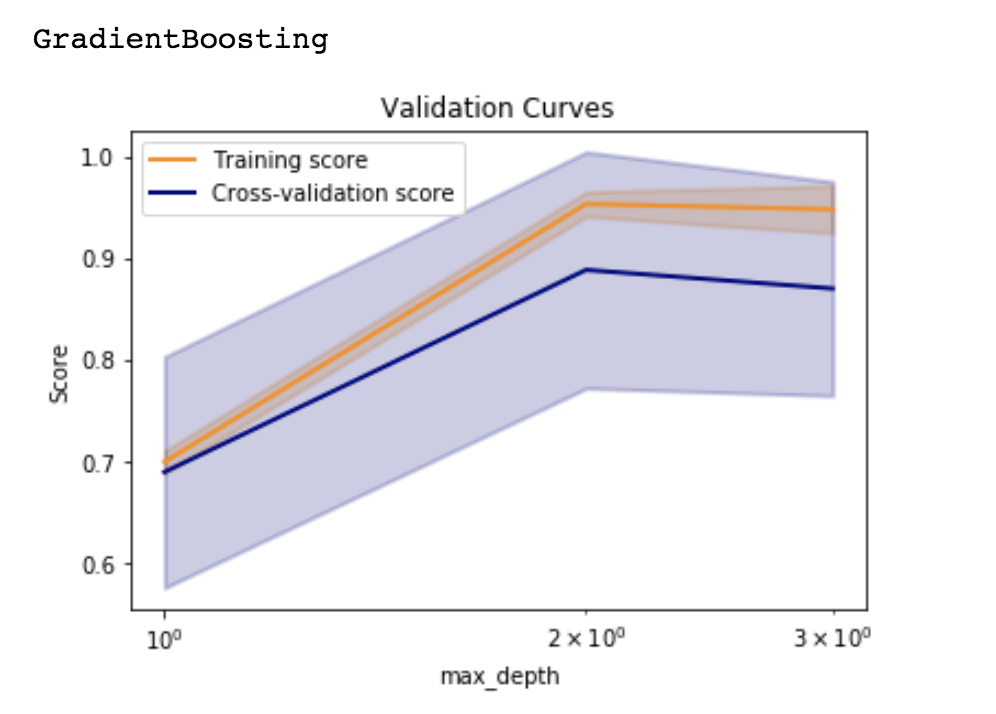


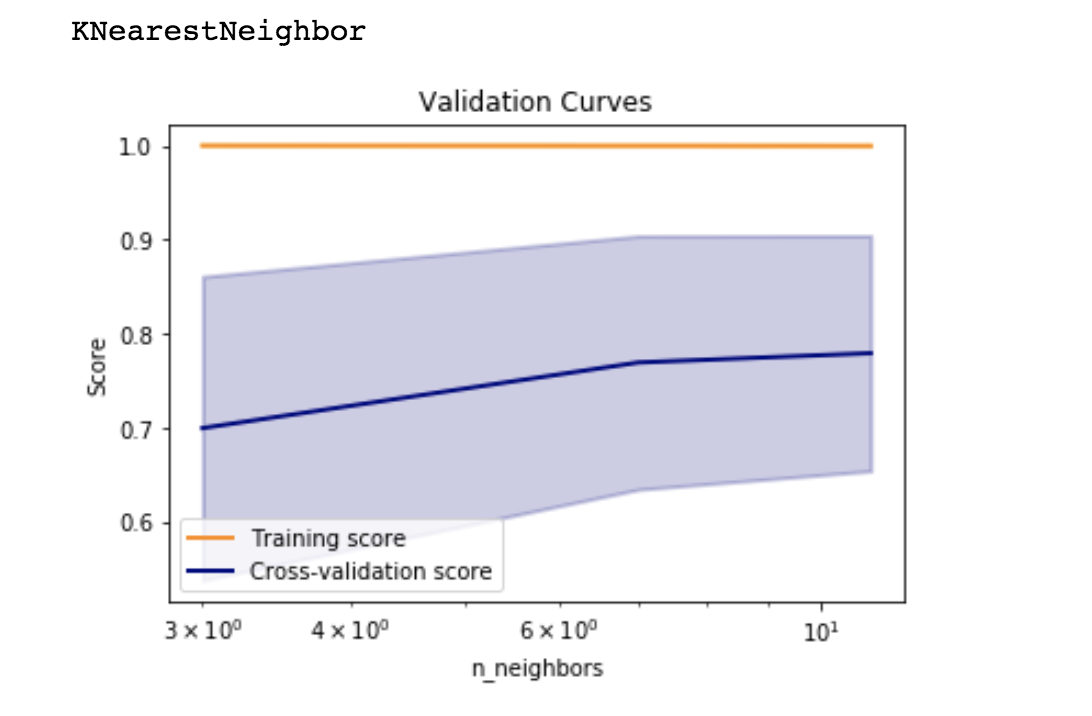
The various classifiers have dramatically different time costs when performing the learning curve and complexity curve valuation. Indeed, different classifiers have different training time – for example, KNN models would have very little time devoted in training, whereas all the other classifiers considered in this implementation have non-nominal training time. More concretely, in terms of comparing the learning curve analysis, the required wall time is increasing in the order of decision tree < 4 x neural network < 1.5 Gradient Boosting < 1.3 x Support Vector Machine < 100 x KNN.

### Complexity analysis

The various classifiers have dramatically different time costs when performing the learning curve and complexity curve valuation. In terms of comparing the complexity curve analysis, the required wall time is increasing in the order of decision tree time < 2 x neural network time < 2.5 x Gradient Boosting < 3 x Support Vector Machine < 17 x KNN.







Remarkably, with the exception of Support Vector Machine, every one of the five classifiers that we have deployed, an approximate 5% gain in performance can be obtained from tuning the model hyper-parameter. For Support vector machine, the performance gain is much greater than 5%. The gain in performance is defined as the increase in area under the Receiver Operating Characteristic curve. More specifically, the performance gain was due to tuning the following hyper-parameters:

* 5% gain based on tuning the maximum depth of decision tree on a decision tree classifier: a pre-pruning method, as opposed to rule post-pruning. The optimal depth of tree for this implementation is 4. The deeper the decision tree is fit, the more likely that training set has been over-fitted and the generalization power therefore decreases.
* 5% gain on tuning the validation fraction on a neural network classifier
* 15% gain on tuning the maximum depth of gradient boosting classifier
* 5% gain on tuning the C on the support vector machine classifier. The value of C controls the smoothness of decision boundary while making correct classifications as much as possible.
* 5% gain on tuning the value of k in KNN classifier

Among the five classifiers being classified, neural network classifier results in the best performance, defined as the largest area under the ROC curve, at approx. 90%. Considering the required training time (compared above), neural network classifier has the optimal combination of performance and training time – not only it has the highest performance based on our valuation metric, it is consistently the second least model in terms of training time complexity of all five classifiers in question.

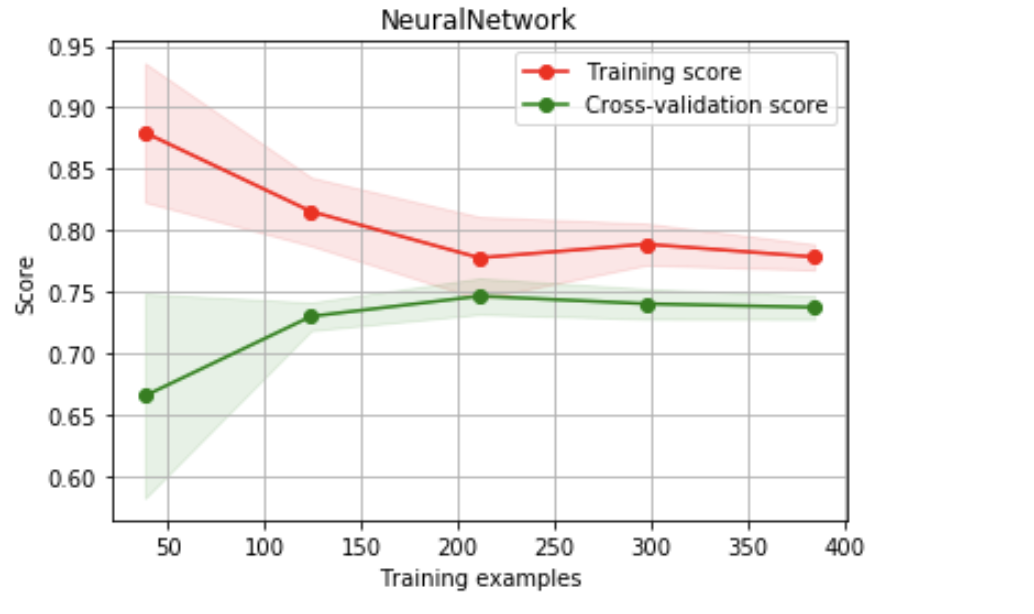
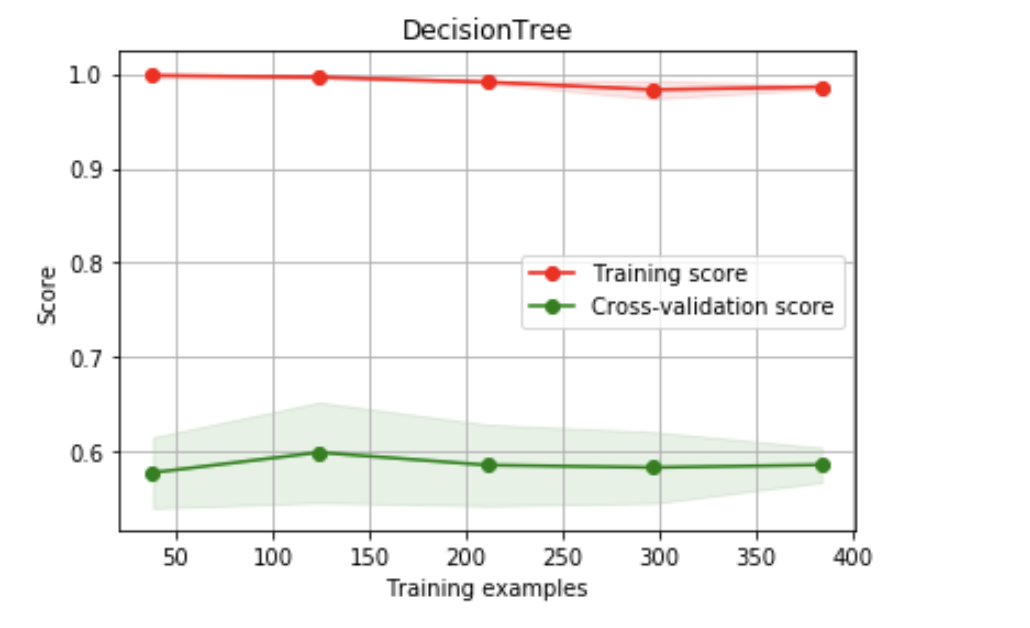
## 

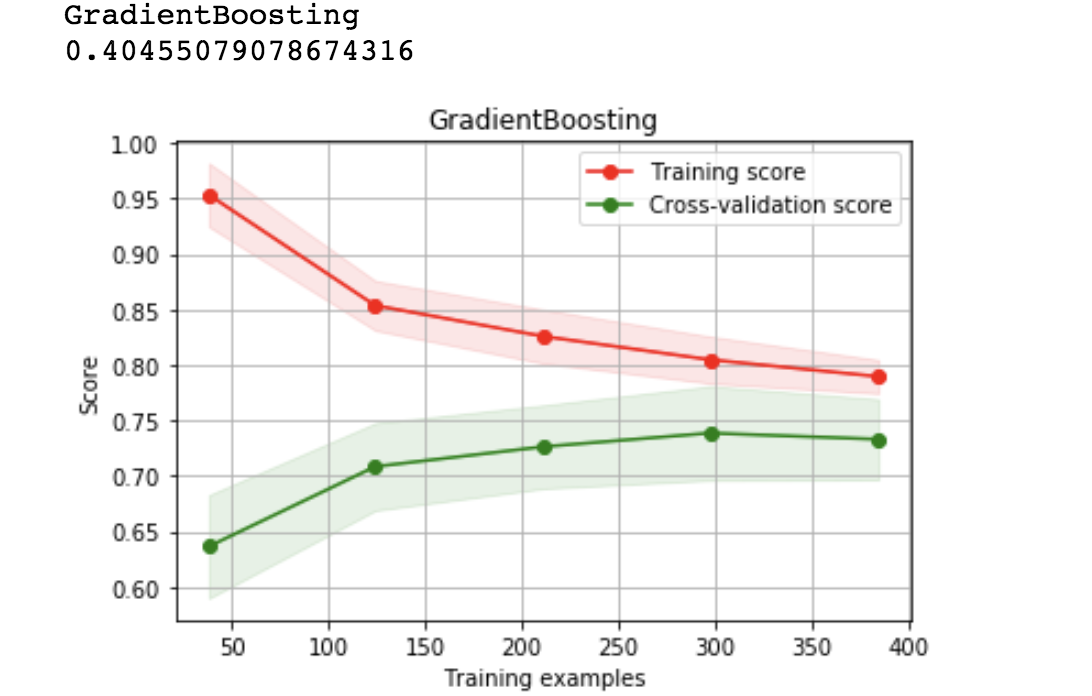
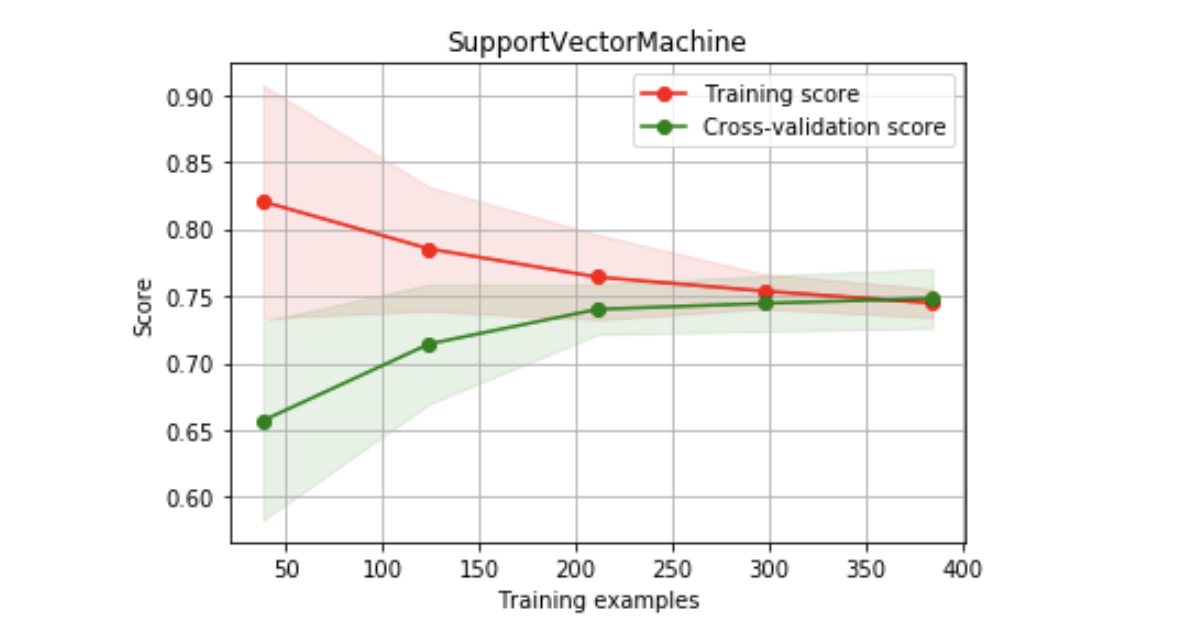
## Blood donation prediction dataset

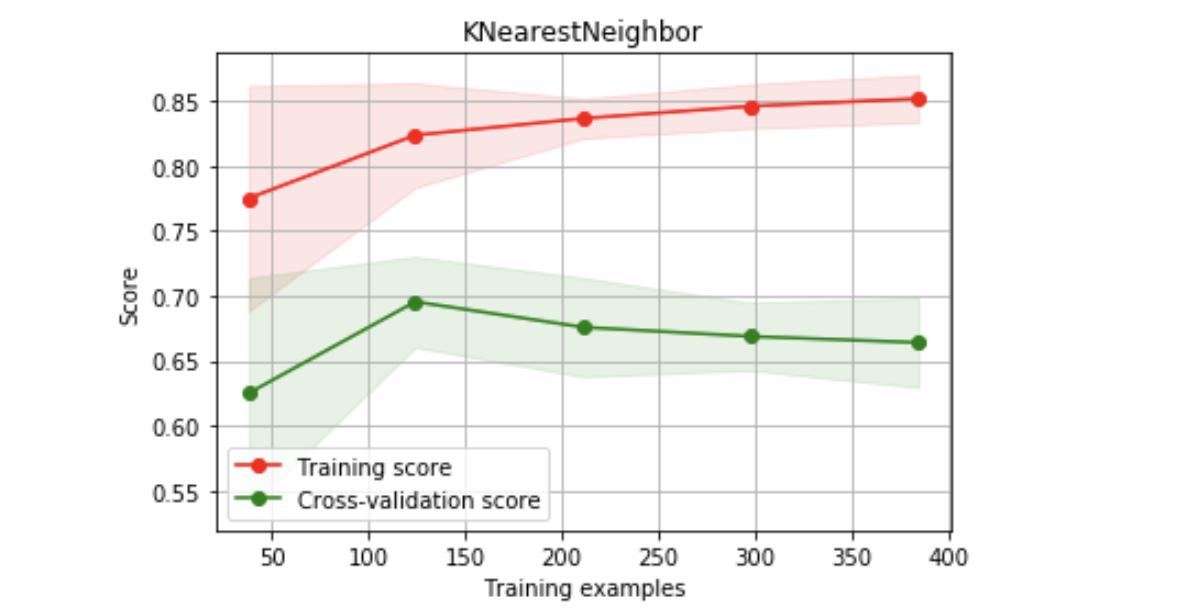
Contrary to the Freddie Mac dataset, the challenge with the Blood Donation dataset is its relative small size and limited feature space.

### Learning curve analysis

Due to its relative small size, the various classifiers have very similar time costs when performing the learning curve and complexity curve valuation. More concretely, in terms of comparing the learning curve analysis, the required wall time is increasing in the order of KNN < Decision Tree < Gradient Boosting < Support Vector Machine < Decision Tree < Gradient Boosting < Gradient Boosting < 4 x Neural Network.

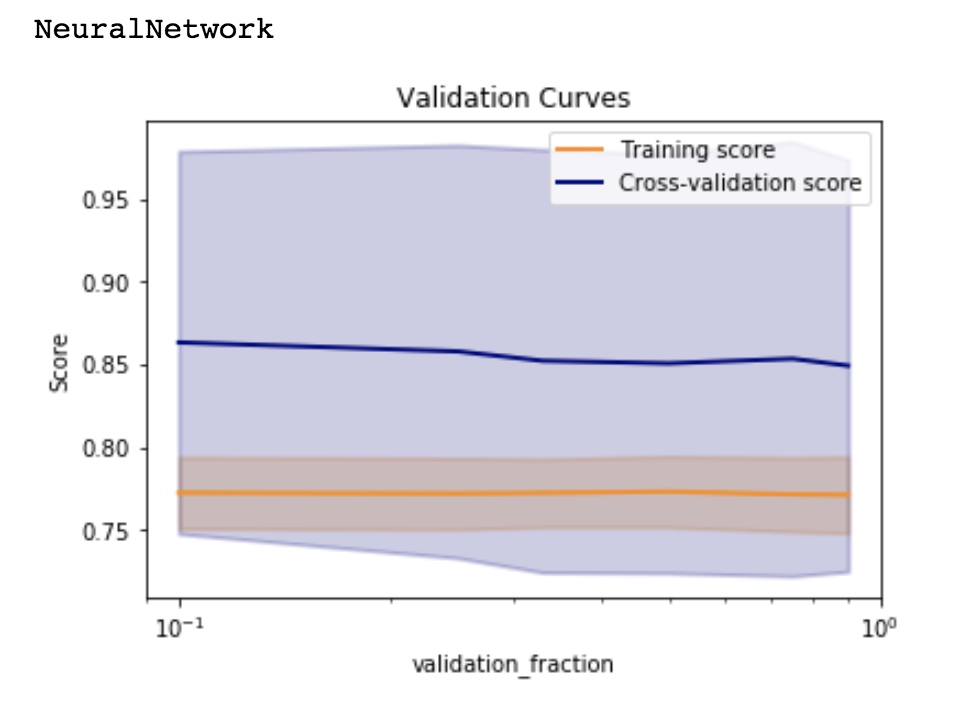
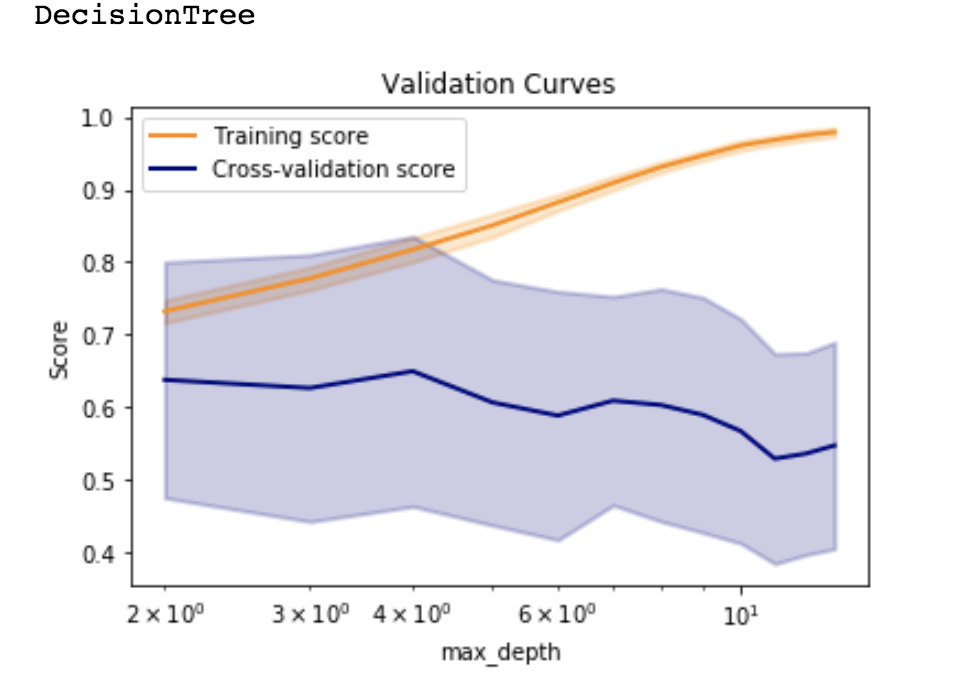


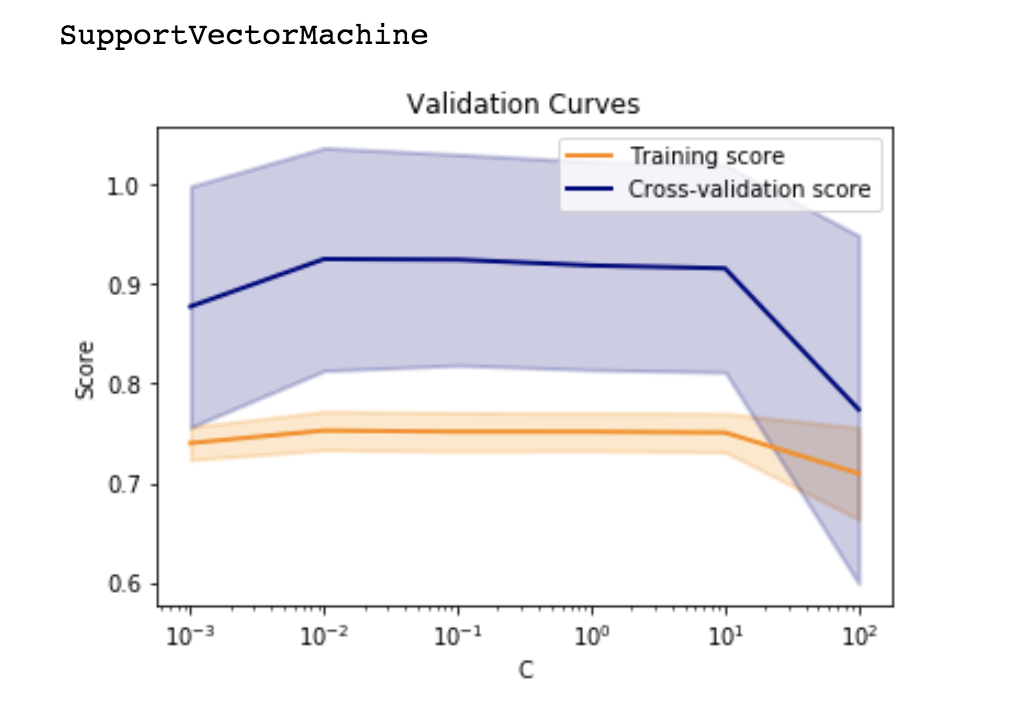
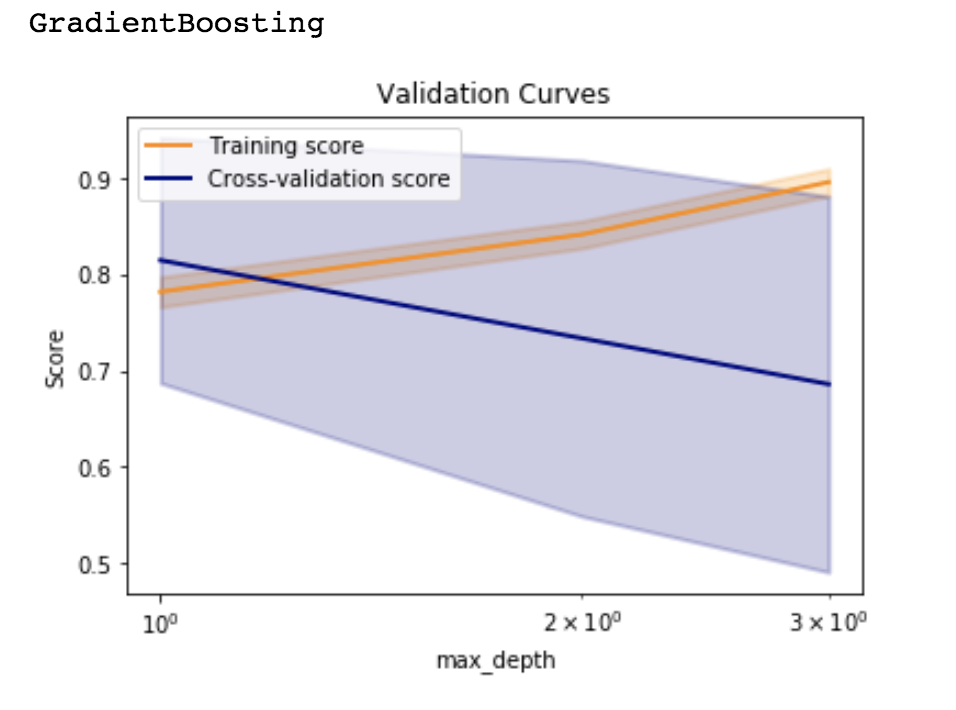


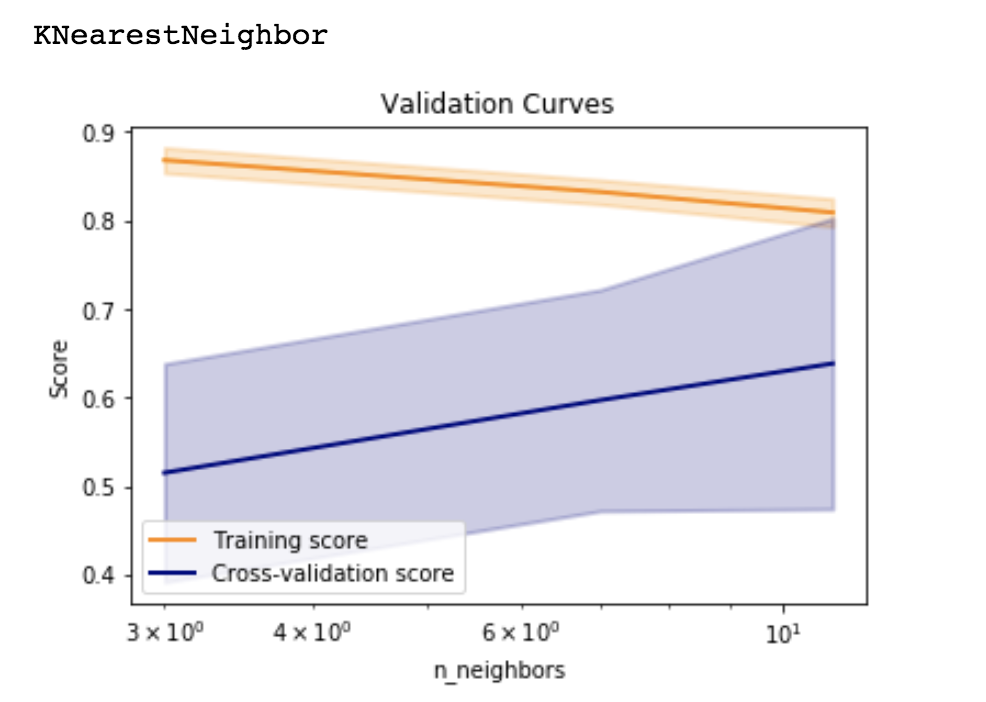


### Complexity analysis

Due to its relative small size, the various classifiers have very similar time costs when performing the learning curve and complexity curve valuation. In terms of comparing the complexity curve analysis, the required wall time is increasing in the order of KNN < 3 x Decision Tree < Gradient Boosting < Support Vector Machine < 4 x Neural Network.





c

Remarkably, with the exception of Support Vector Machine, every one of the five classifiers that we have deployed, an approximate 5% gain in performance can be obtained from tuning the model hyper-parameter. For Support vector machine, the performance gain is much greater than 5%. The gain in performance is defined as the increase in area under the Receiver Operating Characteristic curve. More specifically, the performance gain was due to tuning the following hyper-parameters:

* 5% gain based on tuning the maximum depth of decision tree on a decision tree classifier: a pre-pruning method, as opposed to rule post-pruning. The optimal depth of tree for this implementation is 4. The deeper the decision tree is fit, the more likely that training set has been over-fitted and the generalization power therefore decreases.
* 5% gain on tuning the validation fraction on a neural network classifier
* 15% gain on tuning the maximum depth of gradient boosting classifier
* 5% gain on tuning the C on the support vector machine classifier. The value of C controls the smoothness of decision boundary while making correct classifications as much as possible.
* 5% gain on tuning the value of k in KNN classifier

Among the five classifiers being classified, neural network classifier results in the best performance, defined as the largest area under the ROC curve, at approx. 90%. Considering the required training time (compared above), neural network classifier has the optimal combination of performance and training time – not only it has the highest performance based on our valuation metric, it is consistently the second least model in terms of training time complexity of all five classifiers in question.