CS 7641 Machine Learning - Assignment 1

student: Xinru Lu - xlu320 - xlu320@gatech.edu

1. Abstract

This assignment will focus on some techniques in supervised learning, and their performance across different scenarios.

2. Introduction

Five learning algorithms will be tested and compared:

- Decision trees with some form of pruning
- Neural networks
- Boosting
- Support Vector Machines
- k-nearest neighbors

I will use implementation from `sklearn` package to run the learning process and only compare the results. Even though I will not implement any code, I will explain the selection of function and its implementation briefly, as well as the selection parameters.

3. Dataset

For this project, I selected the following datasets:

- Check Loan Eligibility
- Student Performance (Math only)

The loan eligibility data set is a set of processed data for modeling the eligibility check, based on features like gender, education, income, loan amount, credit history, etc. It consists of 12 columns, of which 10 are integer columns (binary) and 2 are decimal columns. It has a clear `y` column named `Loan_Status` (detailed information can be found in the dataset documentation). I have evaluated it from a ethical perspective in another course at OMSCS (AI Ethics) and it is an interesting data set to evaluate historical biases as well. Therefore, I would like to use it for the ML algorithms and see if different algorithms will induce different biases towards different groups.

The student performance data set was drawn from student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school related features (detailed information can be found in the dataset documentation) and it was collected by using school reports and questionnaires. I have only selected the data of the Math subject. The data was specifically modeled under binary/five-level classification and regression tasks. Interesting point about this data set is that it includes three potential `y` columns: G1, G2, G3, corresponding to 1st, 2nd and 3rd period grades. However, the G2 is dependent on G1, and G3 is dependent on both G1 and G2. I find that it could be useful for Bayesian analysis in the future. Therefore, I would like to use it for the ML course as it is versatile and easy to implement for multiple models.

4. Method

The implementation methods will be briefly introduced in the following section, but many details will be skipped since it is available on `sklearn` documentation page

4.1 Decision trees with some form of pruning

I selected the implemented algorithm from `sklearn.tree.DecisionTreeClassifier`, which has the training part as well as the pruning method (`cost_complexity_pruning_path`). By default, it uses GINI (see reference for calculation) as the criterion to measure the quality of a split. It uses `best` as its strategy to choose the split at each node so that we make sure to pick the attribute with the best GINI coefficient.

Detailed code in file clf_tree.py.

4.2 Neural networks

I selected the implemented algorithm from `sklearn.neural_network.MLPClassifier`. By default, it limits hidden layer size to 100, which is what I will use here. I selected "logistic" as its `activation` parameter, which uses the logistic sigmoid function we discussed in class.

Detailed code in file neural.py.

4.3 Boosting with Bagging

I selected the implemented algorithm from `sklearn.ensemble.BaggingClassifier` as my boosting algorithm for the decision tree model. I selected parameters `max_samples=0.3`, `max_features=0.8` so that for each individual model to be bagged, it will sample 30% of the data and 80% of the features to be considered. I will also use larger alpha values for more aggressive pruning.

Detailed code in file bagged.py.

4.4 Support Vector Machines

I selected the implemented algorithm from `sklearn.svm.SVC` as my SVM algorithm. I selected parameter `gamma='scale'` for ingesting the gamma values to the kernel functions. Other parameters will be kept at default values. For the kernel functions, I will use `'rbf'` and `'sigmoid'` to test the model performance.

Detailed code in file svm.py.

4.5 k-Nearest Neighbors

I selected the implemented algorithm from `sklearn.neighbors.KNeighborsClassifier`

Detailed code in file knn.py.

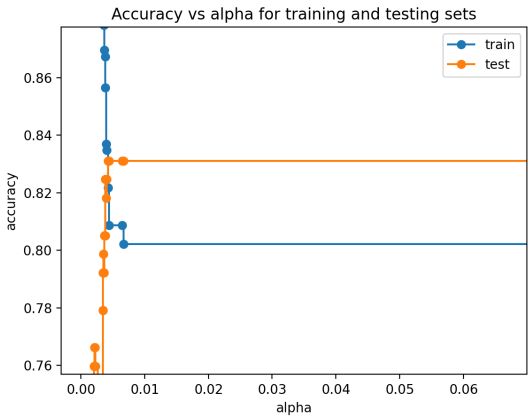
5. Results Discussion

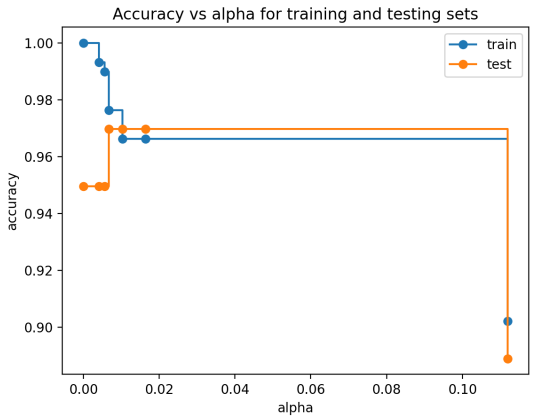
5.1 Decision trees with some form of pruning

Pruning selection:

To select the proper alpha values for the two data sets, I tested all alpha values from the `cost_complexity_pruning_path`, and got the following plots:

- Loan eligibility data set vs Math course data set:



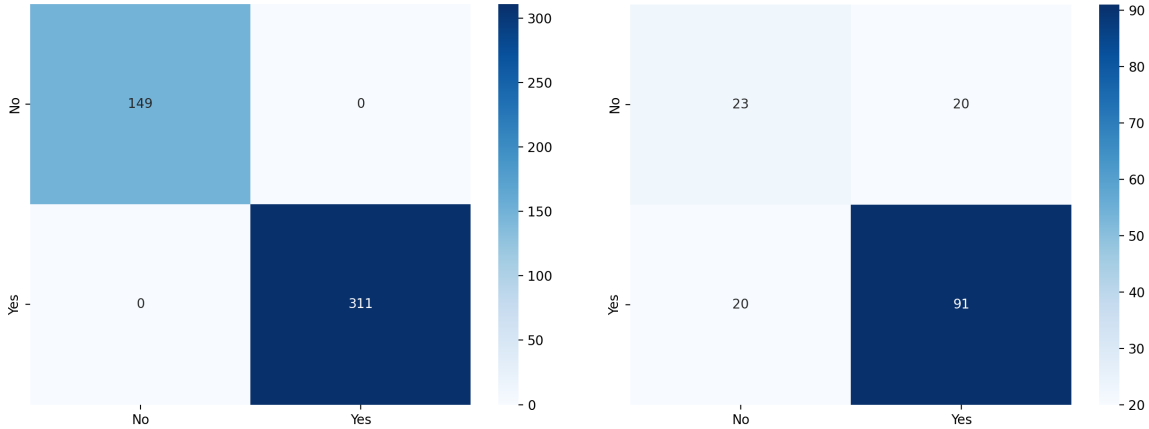


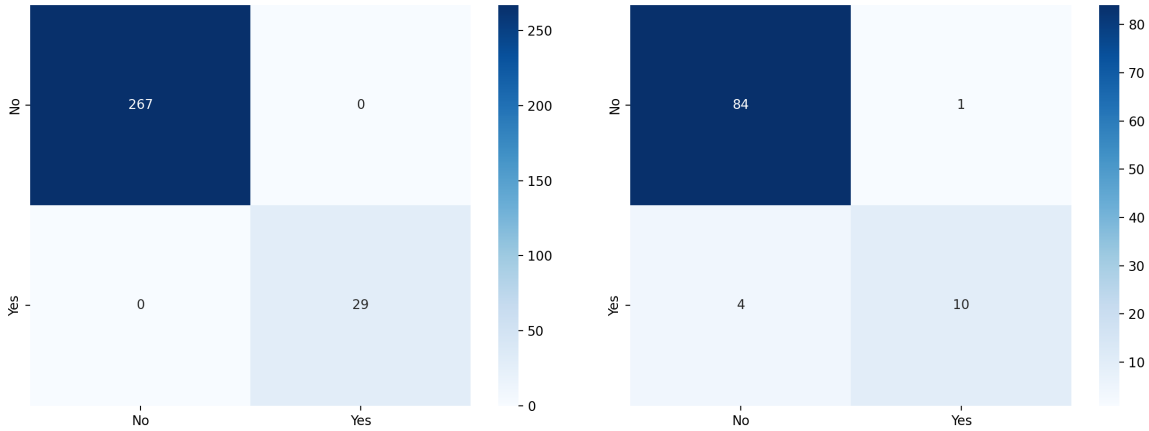
From the above plots, I decide to choose alpha 0.005 and 0.012 respectively, as that is when the training and testing scores are close enough.

Accuracy scores for the decision tree models prior and after pruning:

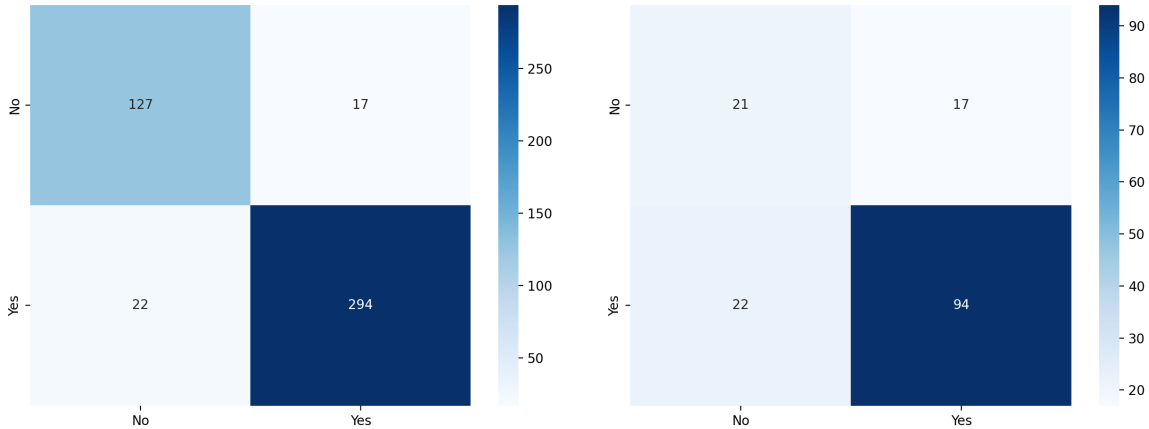
- Without pruning:
  - Loan eligibility data set:
    - Current tree has 209 nodes, with alpha: 0
    - Train score 1.0 (left upper)
    - Test score 0.7402597402597403 (right upper)
  - Math course data set:

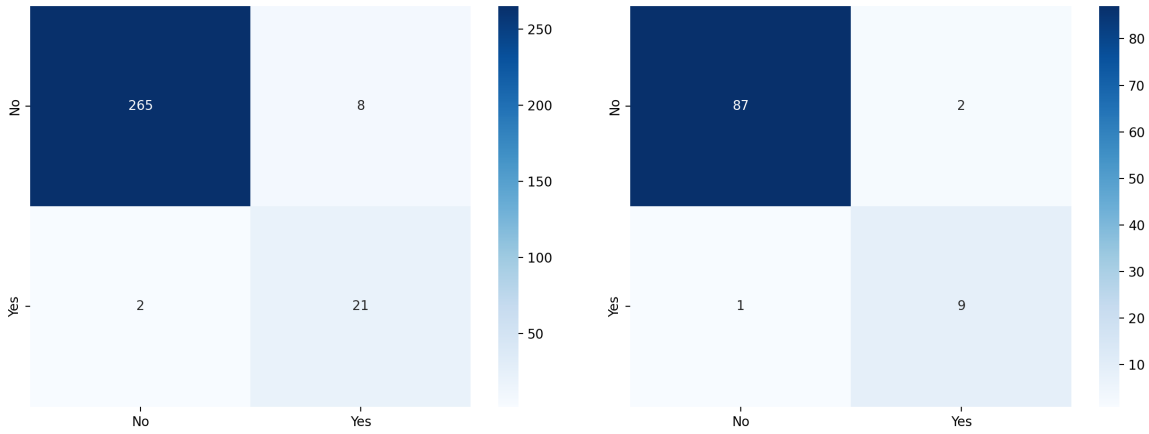
- Current tree has 21 nodes, with alpha: 0
- Train score 1.0 (left lower)
- Test score 0.9494949494949495 (right lower)





- With pruning:
  - Loan eligibility data set:
    - Current tree has 11 nodes, with alpha: 0.005
    - Train score 0.808695652173913 (left upper)
    - Test score 0.8311688311688312 (right upper)
  - Math course data set:
    - Current tree has 5 nodes, with alpha: 0.012
    - Train score 0.9662162162162162 (left lower)
    - Test score 0.9696969696969697 (right lower)



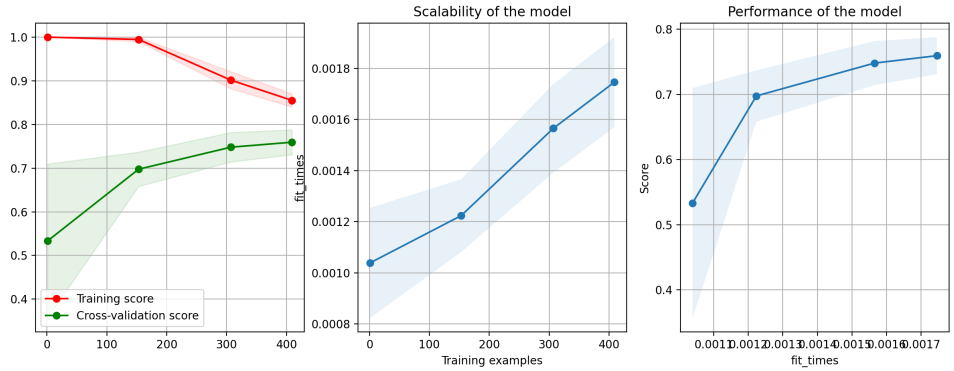


CV:

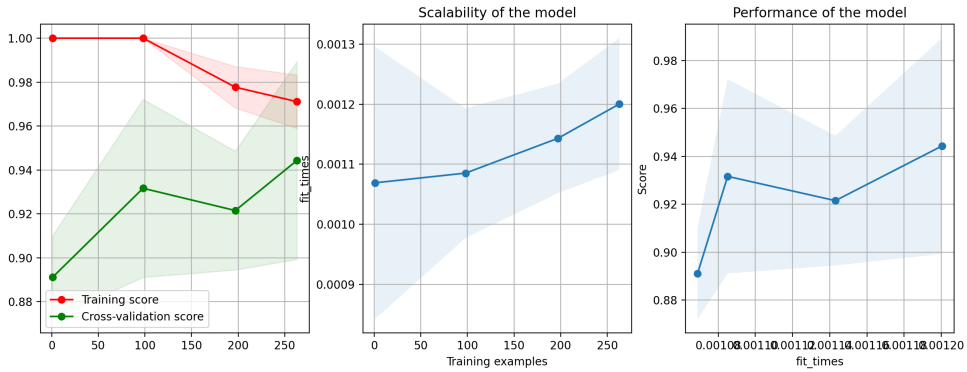
From the above, I observe that there was overfitting when alpha is 0 (no pruning), as the accuracy for training data set is 100%, the testing data set got a significantly lower score. After pruning, despite that the training data set has a slightly lower accuracy, it fits the testing data better than before, which suggests that it will more likely fit real world data better.

Learning curve:

- Loan eligibility data set:



- Math course data set:



We can observe that as training examples increase, the training time for the tree models seems to increase linearly. As it increases, the performance of the model also increases. We also observe that with more training samples, the accuracy of the model increases, as the cross-validation score increases.

5.2 Neural networks

Accuracy scores for the neural network model

- Loan eligibility data set:
  - Train score 0.6934782608695652 (left upper)
  - Test score 0.7272727272727273 (right upper)
- Math course data set:
  - Train score 0.956081081081081 (left lower)

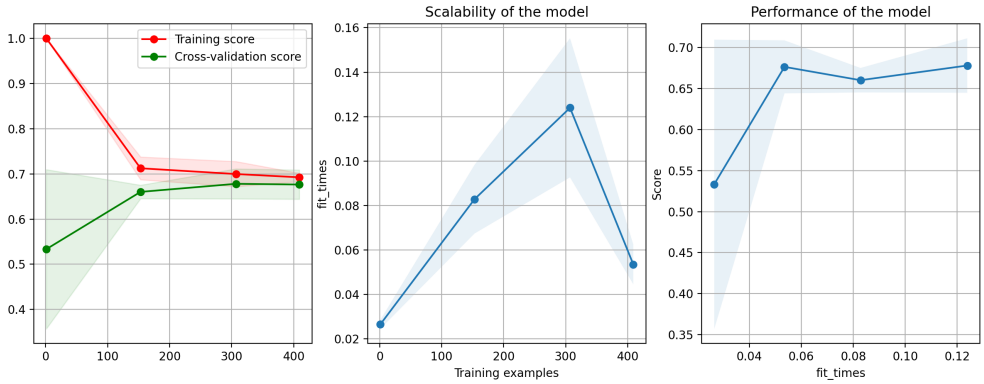
- Test score 0.9292929292929293 (right lower)

CV:

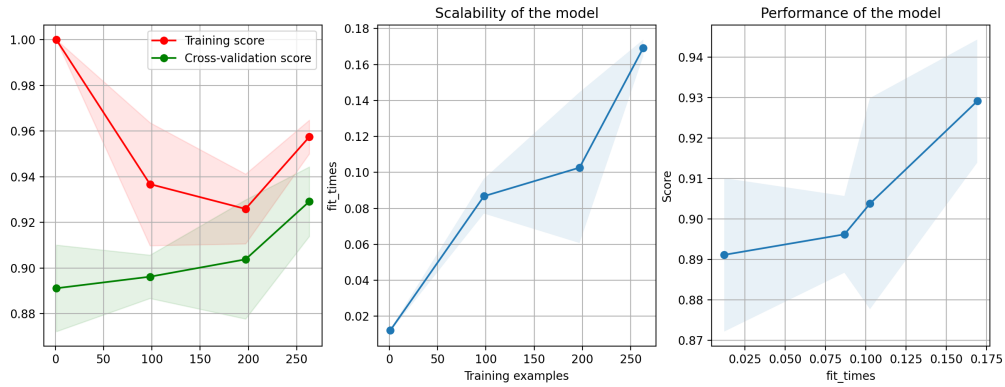
From the above, I observe that there was the model is quite balanced as the test scores are not significantly different from the train scores. Although for loan eligibility test set, the overall accuracy is significantly lower from the classification model.

Learning curve:

- Loan eligibility data set:



- Math course data set:



We can observe that as training examples increase, the training time for the tree models increases as well. The fit_time for sample size around 400 dropped was because the model could not converge and stopped earlier. As it increases, the performance of the model also increases. We also observe that with more training samples, the accuracy of the model increases, as the cross-validation score increases. For loan data set, at some point, it converges and does not seem to increase much more. I think neural network model is quite limited for these two data sets and might not be the optimal model to use.

5.3 Boosting with Bagging

Here I chose to use pruning more aggressively. Compared to the previous alpha values (0.005, 0.012), I decided to use alpha values (0.007, 0.014) for the two data sets respectively.

Accuracy scores for the bagged tree model

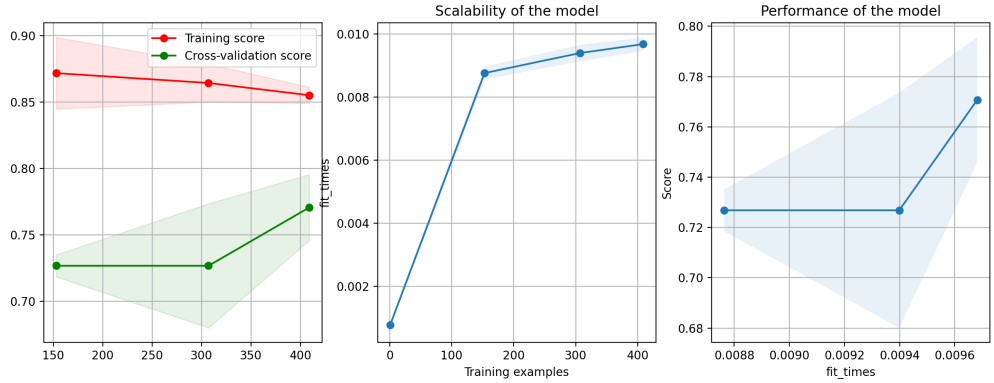
- Loan eligibility data set:
  - alpha value: 0.007
  - Train score 0.808695652173913
  - Test score 0.8246753246753247
- Math course data set:
  - alpha value: 0.014
  - Train score 0.9662162162162162
  - Test score 0.9696969696969697

CV:

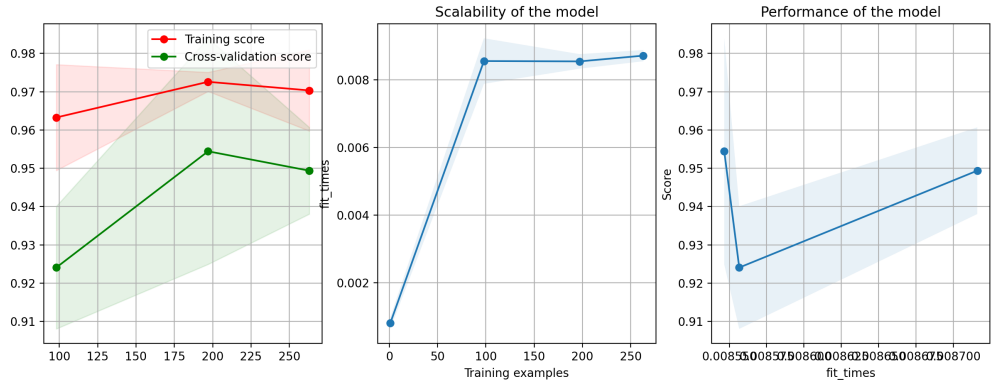
From the above, I observe that there was the model is quite balanced as the test scores are not significantly different from the train scores. Despite more pruning, the accuracy scores are similar to the previous tree model as bagging algorithm improves the performance of the model.

Learning curve:

- Loan eligibility data set:



- Math course data set:



We can observe that as training examples increase, the training time for the tree models increases as well. The overall fit_time is longer than the single tree, however, it is a result of bagging multiple tree models at the same time. The individual training time from the bagged tree will be much shorter than the previous tree. I also observe that the accuracy for the math course data dropped after certain number of samples being fed. It might be a consequence of insufficient data entries.

4.4 Support Vector Machines

Accuracy scores for the bagged tree model

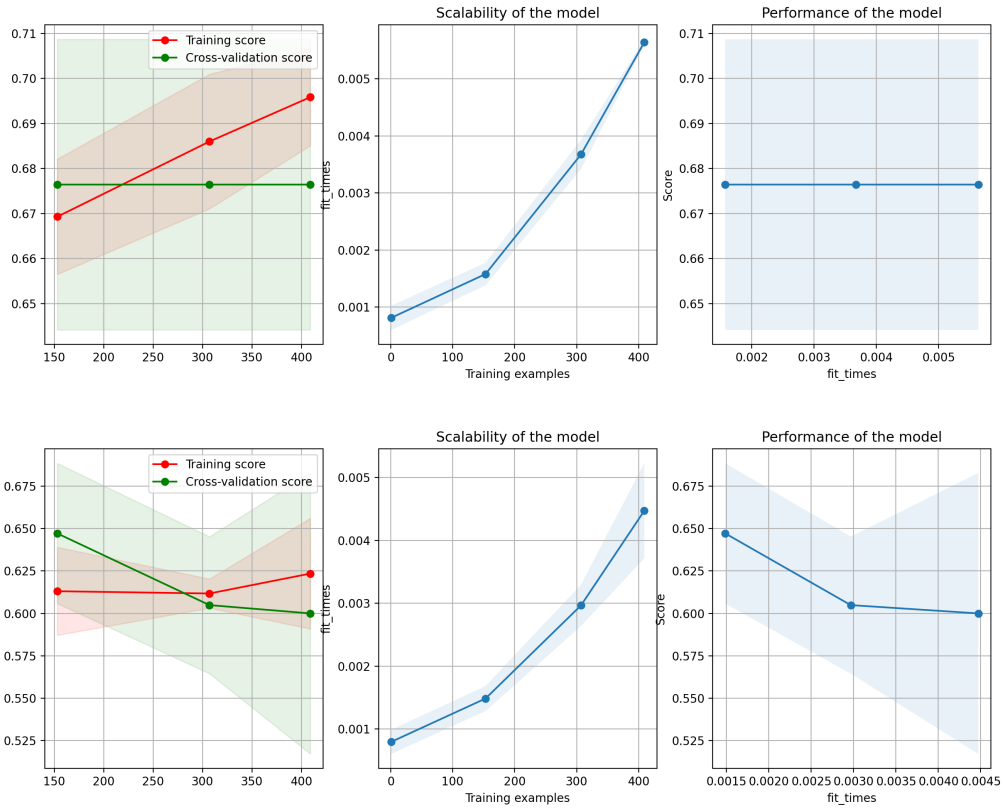
- Loan eligibility data set:
  - RBG model:
    - Train score 0.6804347826086956
    - Test score 0.7207792207792207
  - SIG model:
    - Train score 0.5891304347826087
    - Test score 0.5714285714285714
- Math course data set:
  - RBG model:
    - Train score 0.9391891891891891
    - Test score 0.9595959595959596
  - SIG model:
    - Train score 0.902027027027027
    - Test score 0.8888888888888888

CV:

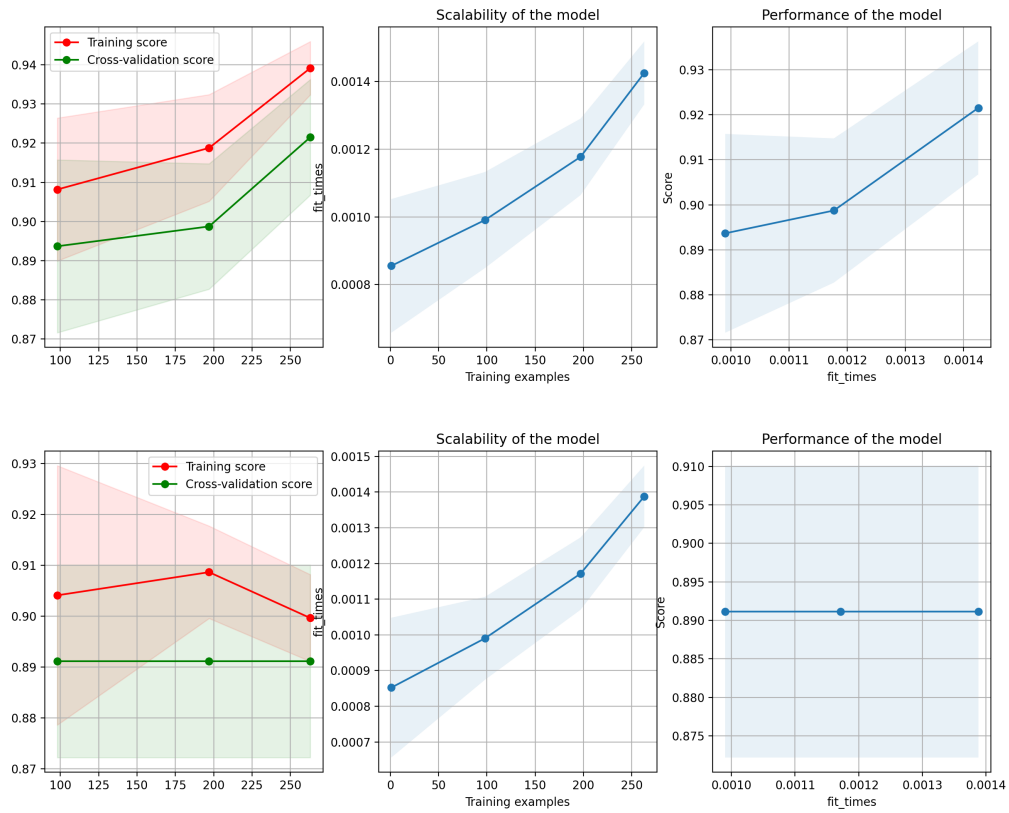
From the above, I observe that there was the model is quite balanced as the test scores are not significantly different from the train scores. RBG kernal function works significantly better than the sigmoid function for both data sets. And I believe it suggests the sigmoid does not apply well to our data sets, as it led to a lower accuracy for both the SVM model and the neural network model.

Learning curve:

- Loan eligibility data set:



- Math course data set:



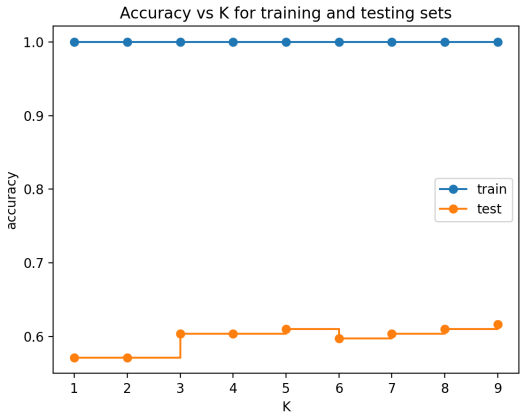
We can observe that as training examples increase, the training time for the tree models increases as well. As RBG function makes more sense here, I would focus on the top image for both data sets (which represents the learning curve for model with RBG kernal function).

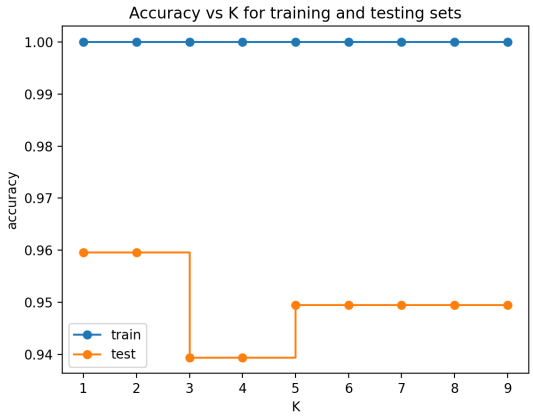
Surprisingly, the performance was not associated with the sample size for loan data set, but they were positively associated for the math score data set. Also I observe that the fit_time increases at a higher rate for SVM model than previous models.

5.5 k-Nearest Neighbors

Test for k values

- Loan eligibility data set vs Math course data set:





From the above, k = 5 seems to improve the model for loan data set. Therefore, I will switch it to k = 5 for loan. I don't see a clear k value that makes the performance the best for math data set, therefore I am sticking with k = 1.

Accuracy scores for the bagged tree model

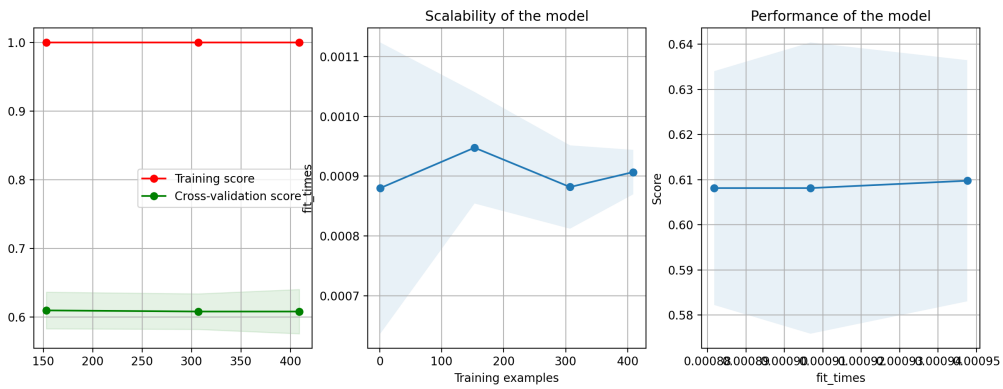
- Loan eligibility data set:
  - Train score 0.808695652173913
  - Test score 0.8246753246753247
- Math course data set:
  - Train score 0.9662162162162162
  - Test score 0.9696969696969697

CV:

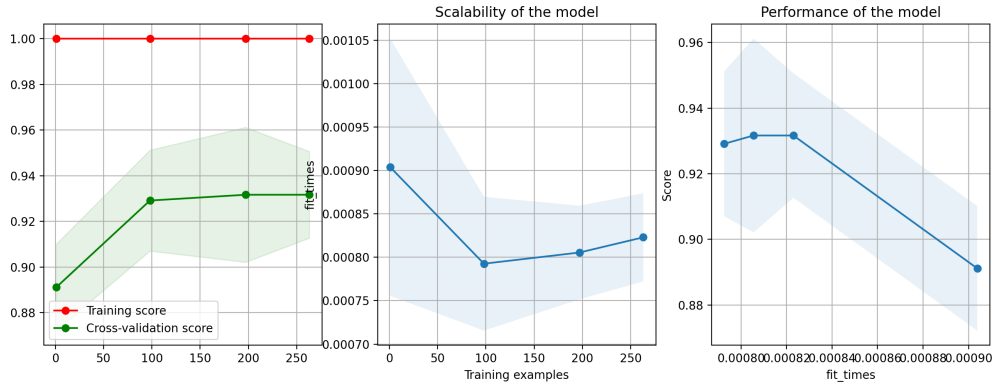
From the above, I observe that there was the model is quite balanced as the test scores are not significantly different from the train scores. The accuracy is generally good for the model.

Learning curve:

- Loan eligibility data set:



- Math course data set:



We can observe that as training examples increase, the training time actually can drop (e.g. from 150 to around 300, the fit_time for the loan data set dropped). It is not very significant, and due to limited data size, it is hard to say predict whether this behavior can occur again with large training examples. However, it is clear that the fit_time for kNN in general is much lower than the previous models, which happen to be one benefit of utilizing kNN model in general.

Surprisingly, the performance was negatively associated with the training time for math course data set here. Also, for loan data set, the number of the training size did not have an significant impact on its accuracy. It seems to be the case for the math course data set as well, at a relatively large training size range.