Assignment 1 Supervised Machine Learning

Zixin (Wendy) Feng
zfeng305@gatech.edu

Abstract—In this paper, we will explore 5 machine-learning techniques by applying them to 2 classification problems.

1 DATA INTRODUCTION AND DATA PROCESSING

1.1 Iris Data

Iris is a big flower category which has 3 species. This dataset[1] measures various numerical properties of each sample, including the length of the sepal, the width of sepal, and the length of the petal, etc. The data is quite balanced and 3 species have a similar amount of data.

This data is interesting because, among 3 species, one of them is linearly separable from the other two, which would predictably make some machine learning algorithms stand out. To help the analysis, I will encode 3 categories 0, 1, and 2. The data is extremely balanced and all categories matter equally. All features are important because the correlation metrics of all of them with the 'class' feature are higher than 0.4.

Another interesting fact is this data only has 120 data points, which is supposed to add difficulty to some ensemble models.

1.2 Hospital Data – Patient Treatment

This dataset is an electronic Health Record prediction collected from a private Hospital in Indonesia[2]. It is predicated on the laboratory test results of patients, including critical metrics like hematocrit and hemoglobin levels. This dataset classifies patients into two categories: those currently receiving care and those who have completed their treatment. Different from the Iris data, it's much larger, with more than 4000 points. Although the dataset has quite a lot of features, only 2 features' correlation with the 'class' feature is above 0.2.

1.3 Analysis Steps

In my research, I initially employed five distinct machine learning models: K-nearest neighbors (KNN), Decision Tree Classifier, Boosted Decision Tree Classifier, Neural Network, and Support Vector Machine (SVM). My objective was to thoroughly analyze the dataset and optimize the hyperparameters for each model. Subsequently, I concluded the report by evaluating and comparing the F1_micro metrics of the top-performing five models. The reason I chose F1 metric

[1] https://www.kaggle.com/datasets/uciml/iris
[2] https://www.kaggle.com/datasets/saurabhshahane/patient-treatment-classification

is that all categories in both datasets matter equally and I would like to access both precision and recall capacity of the model. Besides, the Hospital dataset is pretty unbalanced, and the Iris dataset has multiple classes, which F1_micro can handle pretty well.

1.4 Cross-validation

Cross-validation has been widely used in this study to mitigate overfitting. I trained a model using Iris data and tested it with and without cross-validation and got below accuracy scores:

Non-Cross Validation F1 was derived by splitting the train and testing data randomly with a proportion of 0.8 and 0.2.

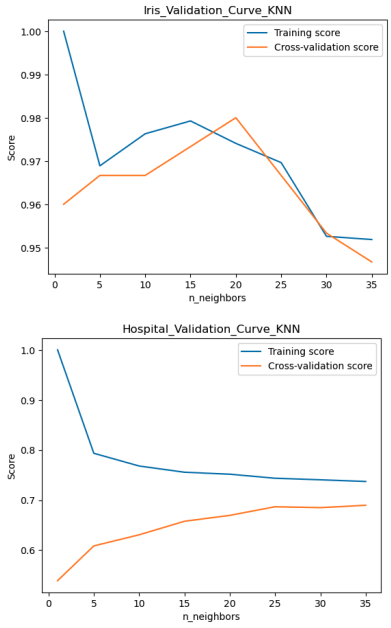
	Cross Validation F1	Non-Cross Validation F1
Decision Tree Model	[1, 1, 0.83, 0.93, 0.8]	0.966
Knn Model	[1, 0.97, 0.9, 0.93, 0.77]	0.967

The above table indicates that non-Cross Validation F1 is between the worst and the best of all cross-validation scores. By using different training and testing combinations, cross-validation could give a more realistic and comprehensive assessment of our model performance. In this report, we used the mean of all cross-validation folds(=5) as the final cross-validation score in the graph.

2 K-NEAREST NEIGHBORS

2.1 hyperparameters

2.1.1 The Number of Neighbors – K

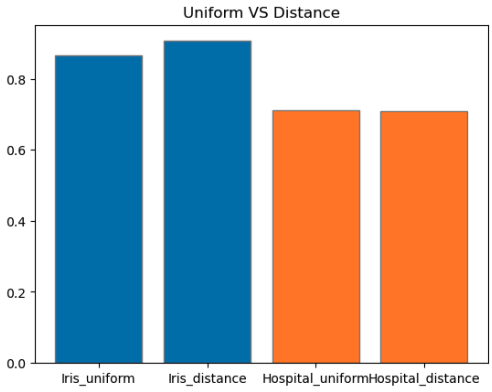


This parameter defines the number of neighbors to be used to decide the class. Let us try different values of K and see how the model performs.

A huge gap between the training and testing curves indicates overfitting. Two graphs on the left show significant overfitting when K is less than 5. The reason is that the KNN model is sensitive to noise when K is small. Also, when K is too small, the model remembers each data point rather than generalizing it, which leads to serious overfitting. However, when K is too large, every point could be in one or two classes, and our model would be oversimplified. When K is near 25, the distance between the

two curves is significantly shorter, and the F1 testing score is still pretty high.

2.1.2 Weighting Strategy

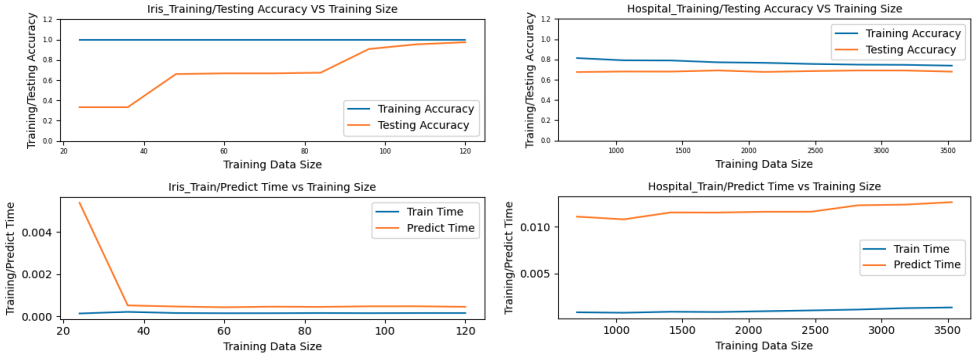


Let us say we have five neighbors who vote to define the cohort. Each neighbor could have a uniform weight in the vote, or the closest neighbor could have the most weight (distance weighting).

Keeping K the same (K=20 for Iris and K=35 for Hospital, as derived from the above 2 graphs), let us see which weighting method performs the best.

Distance weighting gave the Iris dataset the best result. As described in session 1.1.1, one class from Iris data is linearly separable from the other two, which indicates data points from the same class stay closer to each other in this dataset. When close neighbors have a stronger influence, uniform metric performs better. The F1 scores of 2 weighting strategies don't differ a lot in the Hospital dataset.

2.2 Best Model of KNN – Learning curve



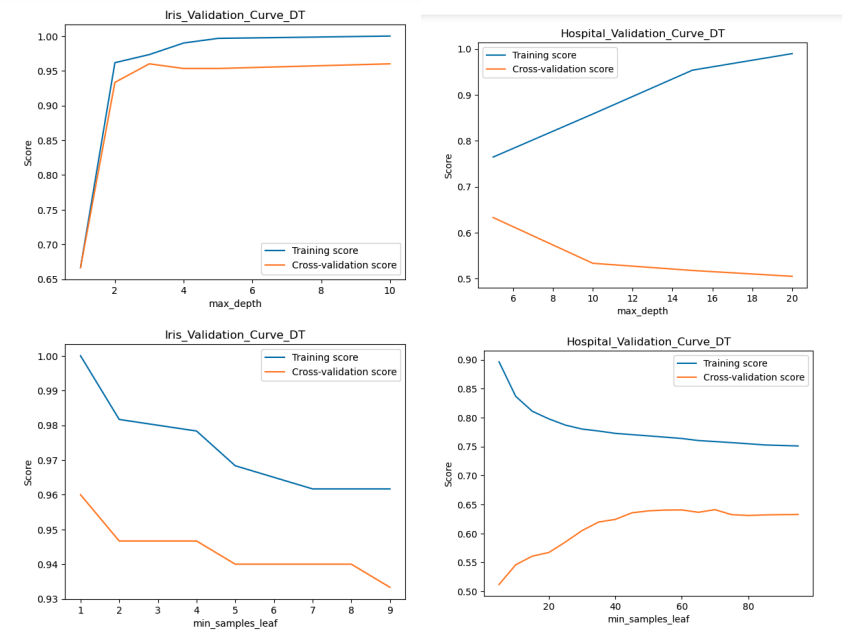
The best KNN model for the Iris data (testing accuracy 0.97) is when K = 15 and weights = "distance". The best KNN model(testing accuracy 0.68) for the Hospital data is when K = 35 and weights = 'uniform'.

In KNN, the prediction time is significantly lower than the training time. It's because, for the KNN model, training is very quick, O(1). With one more train point, the model just needs to store it. No more calculation is needed. However, when it comes to prediction, the model needs to find the closest K point, which took at least O(N * log(N)).

3 DECISION TREE CLASSIFIER

3.1 hyperparameters

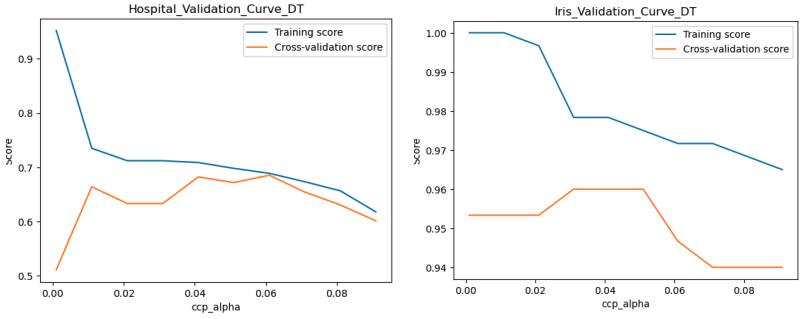
3.1.1 min_samples_leaf and max_depth



Min_sample_leaf is a parameter that sets the minimum number of points a leaf needs to have. As shown in the min_sample_leaf graph of both datasets, when min_sample_leaf is too small, like 1, the tree can keep splitting nodes until the training accuracy reaches 1. However, since the model remembers the data rather than generalizing it, the testing F1 will be bad, and overfitting will appear in this case.

Max_depths serves a similar purpose by setting the maximum depth of a tree to prevent the tree from growing too large. When max_depth is too small in Iris data, both training and testing F1 are very low because the model hasn't learned enough information. In the Hospital dataset, however, when max_depth is 2, the testing score is the highest, which is related to the fact that only a few features have a very high correlation with the class attribute – a few depths are enough to summarize a pretty good set of tree rules. When max_depth is large, like 20 in the Hospital dataset, the tree is too large and the overfitting problem appears. We should keep it in an appropriate range to help generalize data points and reduce overfitting.

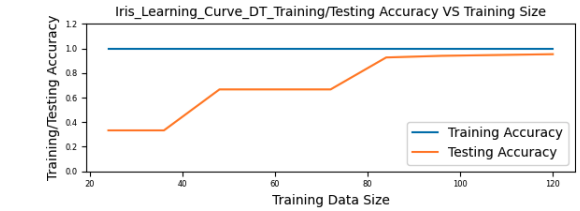
3.1.2 ccp_alpha – Pruning strategy

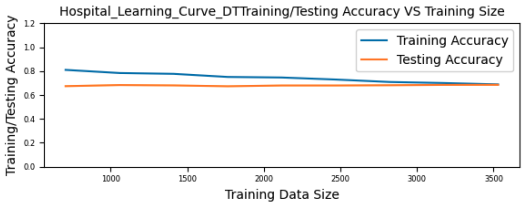


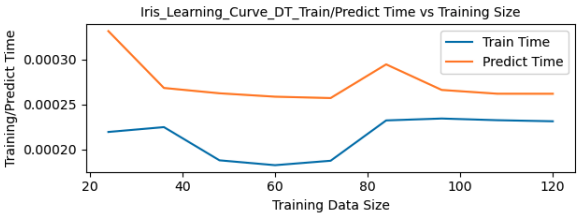
As shown in both graphs, when ccp_alpha gets larger from 0 to 0.05, the training score is closer to the test score and less overfitting happens. This is because ccp_alpha measures if it's worth adding this subtree to the tree based on its impurity. When ccp_alpha is larger, subtrees will be cut more aggressively. However, when ccp_alpha is

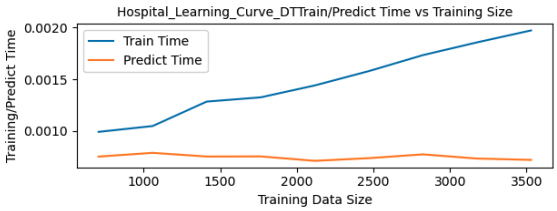
too large, even informative subtrees are cut which hurts model performance rather than improves it.

3.2 Best model – Learning curve









The best DT model for Iris data (testing accuracy 0.95) is when ccp_alpha=0.011, criterion='entropy', max_depth=10, min_samples_leaf=1. The best DT model for Hospital data (testing accuracy 0.68) is when ccp_alpha=0.001, max_depth=1, and min_samples_leaf=5.

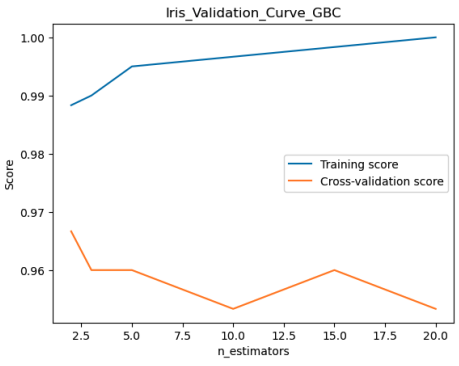
We can see the train time is significantly higher than the predicted time in the Hospital graphs. Why is that? In the decision tree model, constructing the tree takes a lot of time. A simple version of the DT model needs to figure out what is the most informative feature and split data points. Once the model is built, predicting just needs to follow the path and find the result so it's much quicker than training. However, the train time is constantly a little bit slower than the predicted time in the Iris graph. This might imply that the tree is getting too large for such a small amount of data points.

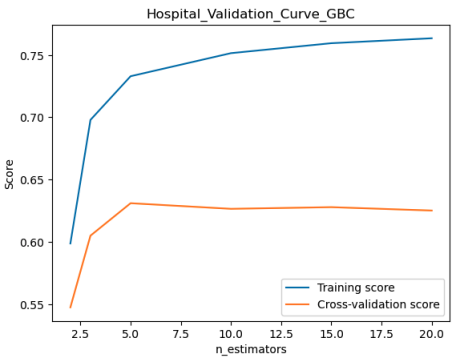
4 BOOSTING MODEL

4.1 Hyperparameters

Boosting is a great ensemble method that combines the result of multiple weak learners into a strong learner. We will use Gradient boosting which builds DT models sequentially.

4.1.1 The Number of weak learners – N

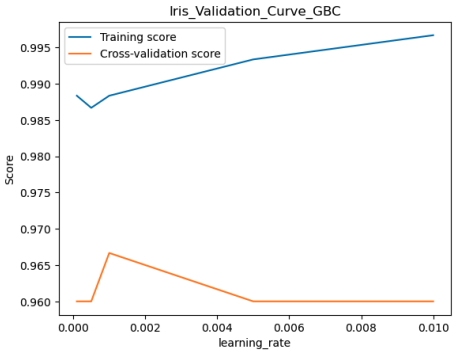


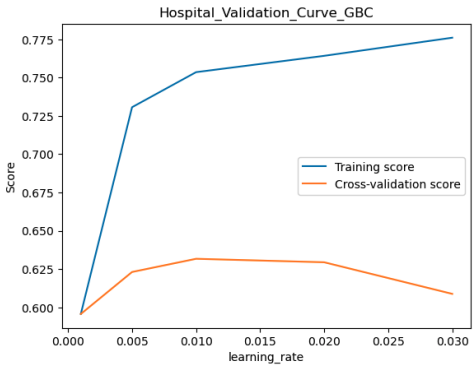


When N increases from 0 to 5 in both graphs, the F1 score increases. This is because more weak learners could help analyze more difficult

cases. In the boosting model, if a point was misclassified in a previous DT model, it will gain more weight in the next DT model's training process. This point will have a larger chance of being correctly labeled in the next DT mode if there is the next one. However, a very large N can add a lot to the train time without adding a lot to the model performance. There will also be overfitting with a large N, as observed from the hospital graph, when n is larger than 5, a significant overfitting appears.

4.1.2 Learning Rate

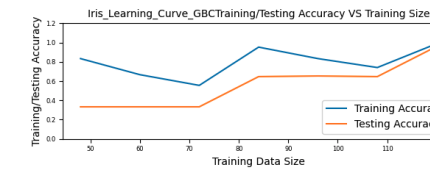


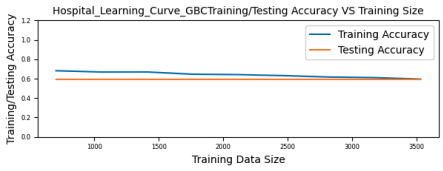


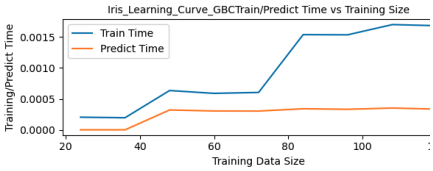
The learning rate defines how much weight we will add to the precious wrongly classified points.

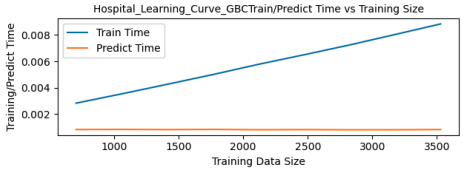
In both graphs, when the learning rate gets very large, there is a serious overfitting problem. This is because when the learning rate is too large, the convergence happens too quickly and we fit data too closely by assigning too much weights to difficult cases. In other words, we overly rely on difficult points. When the learning rate is very small, we will need more time to get convergence, and a larger N is needed.

4.2 Best Model – Learning curve





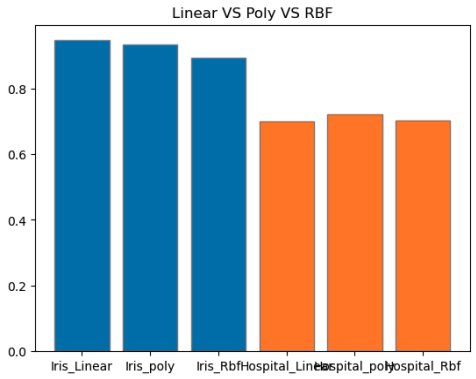




The best-boosting model for the Iris data (testing accuracy 0.96) is when n=2, and the learning rate is 0.015. The best-boosting model for Hospital data (testing accuracy 0.59) is when n = 3, and the learning rate is 0.001. Both have not been improved a lot compared to their underlying decision models, which indicates the first several underlying decision tree models are already well-trained and predict very well. Not a lot of improvement could be made. Our optional N is a small number, which is consistent with my hypothesis.

5 SUPPORT VECTOR MACHINES

5.1 Hyperparameters

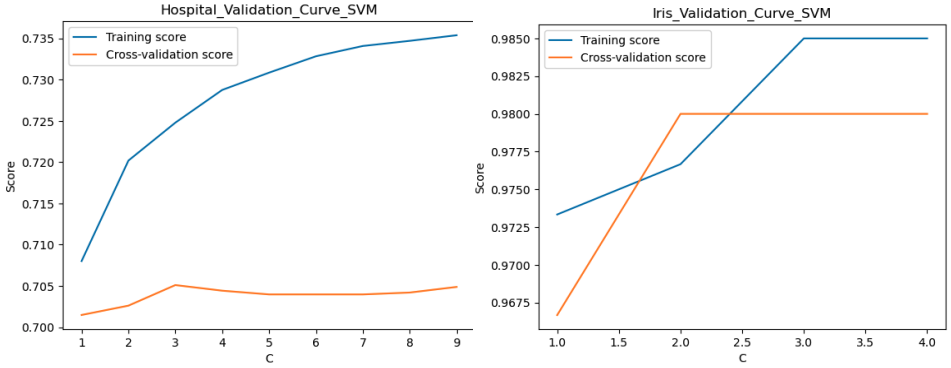


5.1.1 Kernel Type

Based on differences in data distribution, we can choose different kernel types. For the Iris data, since it's linearly separable, 'linear' performs slightly better(2%) than other methods.

'Linear' is the worst for the Hospital dataset because both Poly and RBF can best handle cases that don't fit into linear models.

5.1.2 Regularization Parameter – C



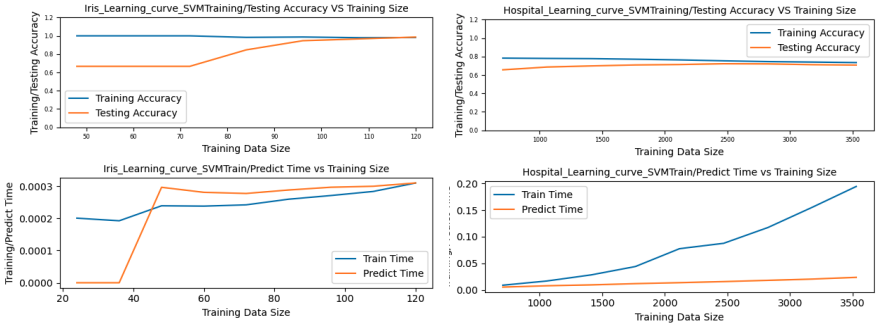
We can see in both graphs, when C increases from 0 to 2, the training and testing F1 scores first get closer and then farther away from each other. When C is very large, there is obvious overfitting.

This is because when C is small, the SVM model boundary will be less sensitive to individual points, which results in more classification errors. Testing accuracy even gets higher than training accuracy in the Iris example.

When C is a large number, the model will take care of each point, which results in higher training accuracy but a higher probability of overfitting because the model doesn't generalize.

When we choose an appropriate C value, (2 in the Iris case and 3 in the Hospital case), the model neither overfits nor makes too many classification mistakes.

5.2 Best Model – Learning curve



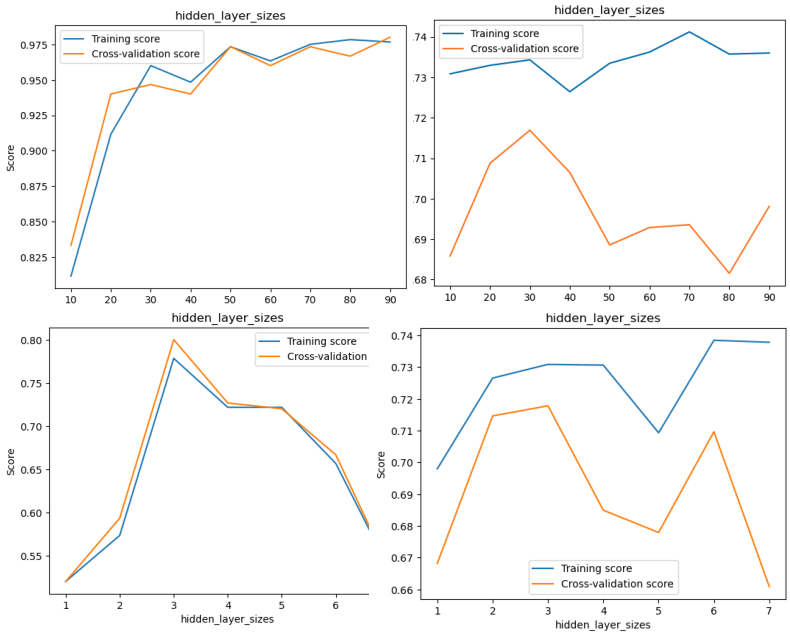
The best SVM model for the Iris data (testing accuracy 0.99) is when the kernel type is linear, and C is 5. The best SVM model for Hospital data (testing accuracy 0.71) is when the kernel type is poly, and C is 2.

According to the graph, the training time of SVM model is usually longer than its prediction time. The gap gets larger when the data size is increasing. This is because SVM uses all datasets in training but only uses a few so-called SV points which define the hyperplane for predicting.

5 NEURAL NETWORK

5.1 Hyperparameters

5.1.1 Number of Hidden Layers and Neurons



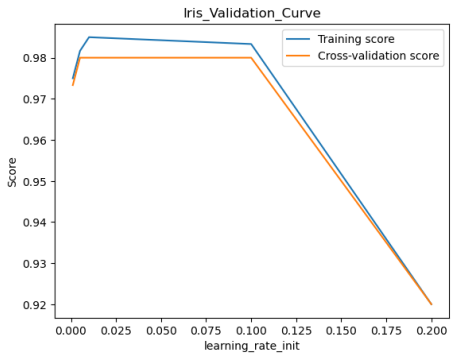
I would like to try different types of layers and Neurons combinations. There are 1, 2 or 3 layers with different numbers of Neurons. Let us first check the result of 1 layer with different neurons. (Two upper left graphs)

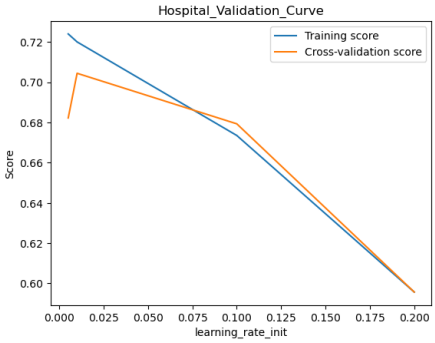
It's clear to see that a large dataset needs more neutrals and a small neutral size is okay for a small dataset. When the number of neurons is too many for the data, we observed overfitting in the Hospital example.

Now let us check the result of 1 -7 layers, each of which contains 5 neurons. (Two lower-left graphs)

With more layers, the model performance could be improved. However, when there are too many layers, the model might overfit, as shown in the Hospital example when the number of layers is higher than 6.

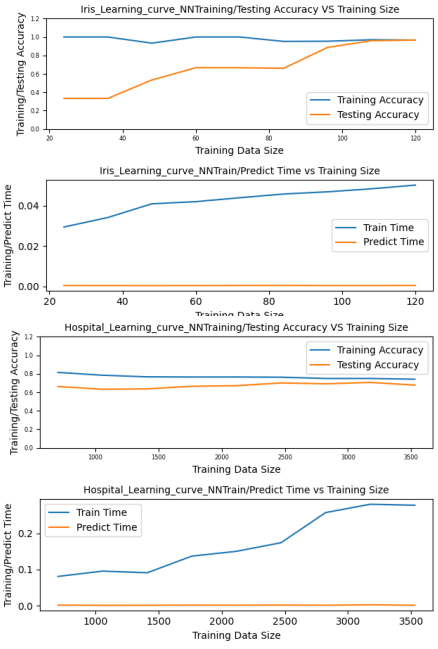
5.1.2 Learning Rate

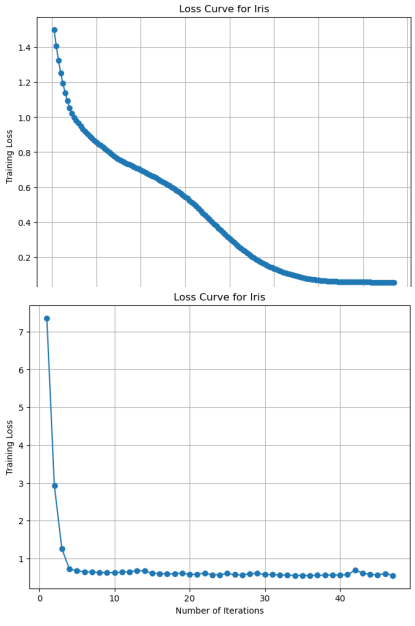




In neutral network models, weights are adjusted based on the learning rate. I tried 0.0001, 0.001, 0.01 and 0.1. In the left graph, both training and predicting accuracy increase when the learning rate is very small (<0.025), and decrease when the learning rate gets higher than 0.1, which indicates a 0.1 learning rate makes the model converge too fast, and we miss the local minimum error point.

5.2 Best Model – Learning Curve and Loss Curve





The best Neutral Network model for the Iris data (testing accuracy 0.97) is when there are 4 hidden layers, each of which has 10 eurons, and the learning rate is 0.005. The best Neutral

Network model for the Hospital data (testing accuracy 0.678) is when there are only 1layers, which has 10 Neurons, and the learning rate is 0.01.

Why does the Hospital data have a larger size but require fewer numbers of layers? The loss curves indicate that in a balanced dataset (Iris), the convergence happens slowly, while in the Hospital case, the model generalizes to unseen data very quickly(when iteration = 5). This is consistent with our assumption that a few features in the Hospital data define the class.

6 CROSS MODEL COMPARISON

6.1 Model Performance Comparison:

	Iris Traning F1	Iris Testing F1	Hospital Traning F1	Hospital Testing F1
Decision Tree	0.9716	0.9464	0.688	0.68515
Boosting	0.9717	0.9465	0.5956	0.595
Neutral Network	0.9816	0.980	0.734	0.705
KNN	1	0.973	0.739	0.679
SVM	0.98166	0.986	0.734	0.707

1) The compatibility of model methodology and data matters:

   Since Iris data has one class that is linearly independent from the other 2 classes, any machine models whose methodology is related to linear models have slightly better performance than the other models, according to the above table. For example, each neutron in the linear Neutral Network model predicts the result by doing a linear transformation of all input features. SVM(kernel =linear) aims to find a linear decision boundary that separates the data. Those models' internal idea is compatible with the data.

2) The boosting model performance of both datasets is not significantly better than the single decision tree model. There are multiple potential reasons: For the Iris data, the dataset is small, and boosting requires a large dataset to effectively train the ensemble. For the Hospital data, weak learners in the boosting models are too weak, and boosting is not able to improve their performance.