Supervised Learning Report

Dataset: Credit Approval

Data (crx.data) can be found in the link: https://archive.ics.uci.edu/ml/machine-learning-databases/credit-screening/

Description of Classification problems:

In the credit approval data, it contains applicants' demographic data, some credit-related data and the credit card approval results. This data set is very useful for bank companies to train the model to predict whether to approve the credit card for new applicants. The data has 10 columns – 9 features and 1 target variables.

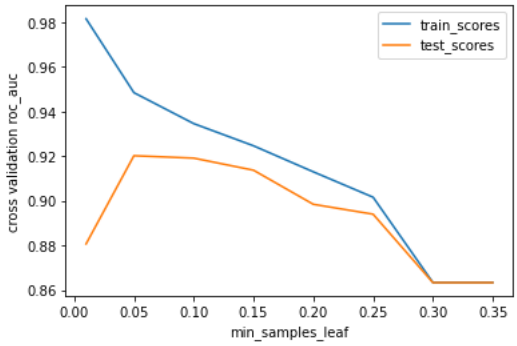
Why data is interesting:

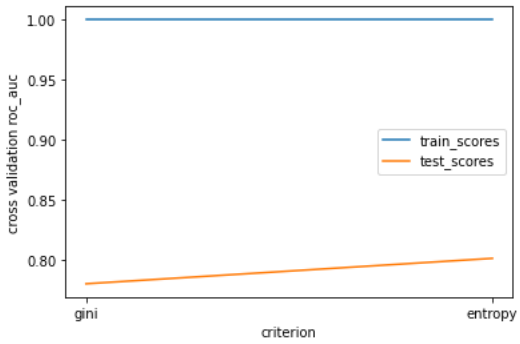
It is a very interesting data set because it has a mix of categorical data and numeric data. The features include applicants' gender, age, debt amount, number of years of employment, prior default flag, employment flag, credit score, and flag of having driver license The target variable is "approved" or "not approved". The data is balanced. This is a binary classification problem, and we can try different classifier on it to see which one can deliver the best result and avoid overfitting and underfitting issues.

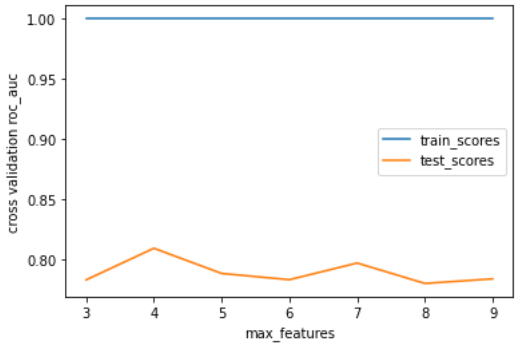
Parameter Tuning Analysis:

I applied 5 algorithms on the dataset. For each algorithm, I tried to tune parameters and see how the model performance changes. Then I use cross validation to select best combination of parameters which contribute to best model. Based on the best model, I plot the learning curve as a function of training data set to do analysis.

1. **Decision Tree**



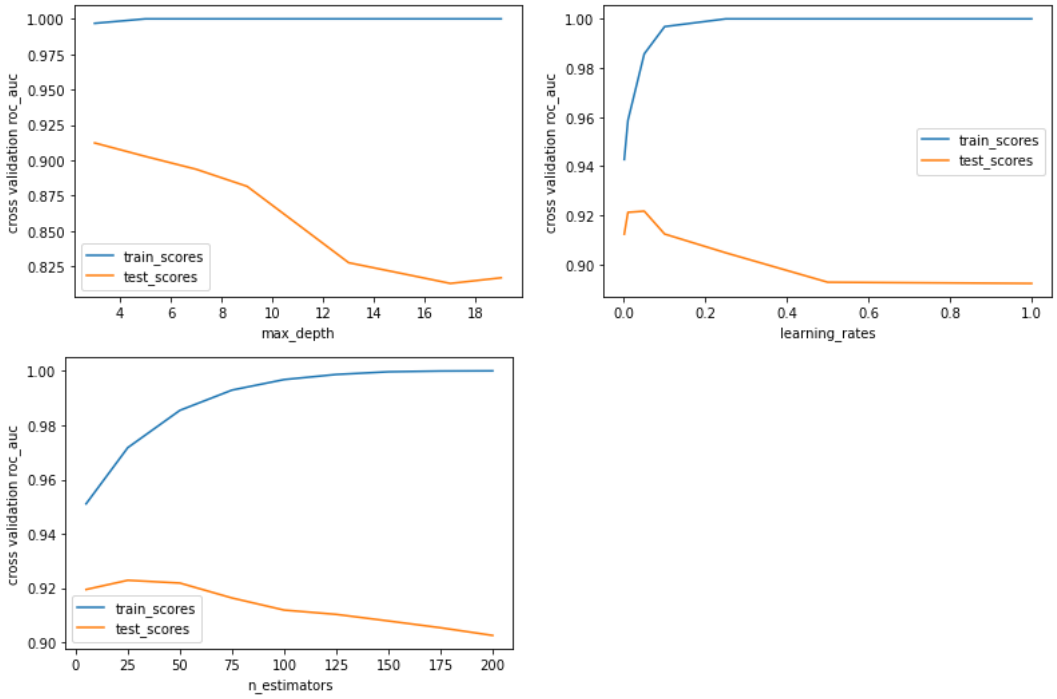




Decision tree is to split the data based on the information gain and then we can prune the tree by limiting depth, minimal samples in the leaf and etc,. Based on the graph above, I think min_sample_leaf = 0.05 is the best choice because at that point,

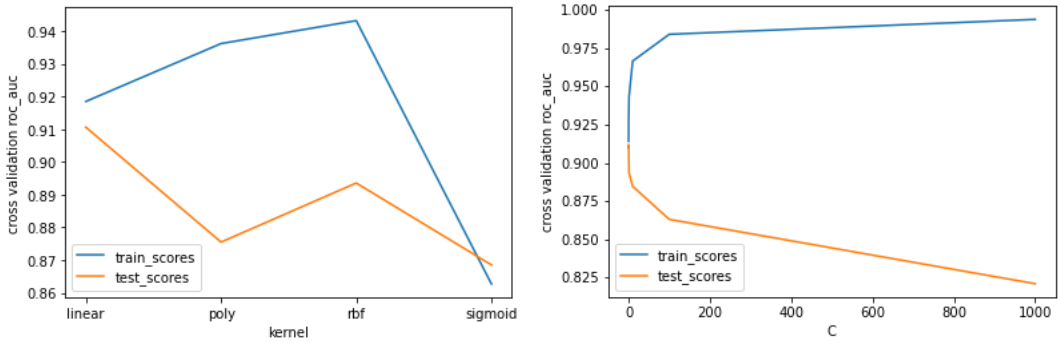
2 lines are close and testing auc roc hit the peak. If we choose higher min_sample_leaf, it would be underfitting because both training and testing scores are going down. So high min_sample_leaf would result in high bias. That's why I do the pruning to make min_sample_leaf equal to 0.05. Choosing gini or entropy would not make any difference because their training scores stay same and testing scores doesn't change a lot. For the max_features, training scores stay same and testing scores fluctuate between 0.75 and 0.8. I think changing max_features cannot help model become better.

2. Gradient Boosting



Gradient boosting model uses boosting method to correct the error in each tree with gradient decent to quickly minimize the loss. We can control the depth to prune the tree. Based on the graphs above, I think max_depth=3 is the best because testing scores decreases, and training scores stay high and stable. With max_depth goes up, the model faces the high variance problem, overfitting the data. In the second graph,learning rate = 0.07 is the best choice because after 0.07, training scores increase and testing scores decrease, resulting in overfitting issue. Also, the same overfitting issue happens in the last graph, so n_estimator= 50 is a good choice.

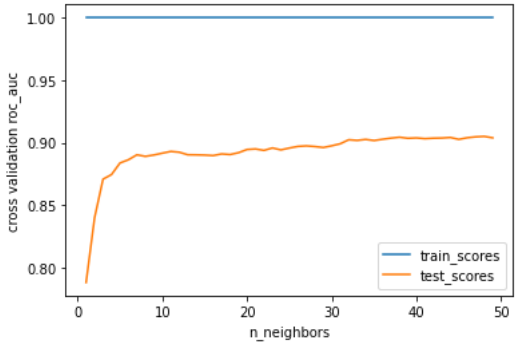
3. SVM

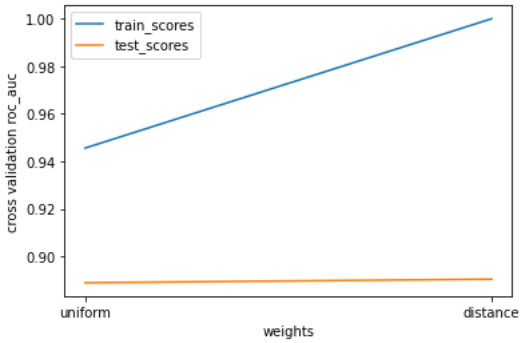


SVM model can find a hyperplane to separate the data. In the process, SVM tries to minimize the error and also masximizr the margin. So we need to find a tradeoff between error rate and margin. If margin is too small, we have low error rate but we may also face overfitting risk. If margin is too large, we have high error and we may face underfitting risk. Based on the graphs above, I think linear kernel performs better because the training roc auc and testing roc auc are very close and both are relatively high. RBF and poly kernel have higher training roc auc but their testing roc auc are much lower so they might have overfitting risk. C is the penalty parameter of the error term. It controls the tradeoff between smooth decision boundary and

classifying the training points correctly. In the second graph, when C is smaller, train and test roc auc are closer. When C is higher, the model has low bias and high variance. So I prefer to choose C=0.001

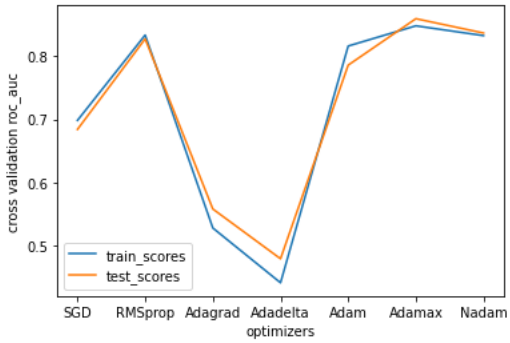
4. KNN

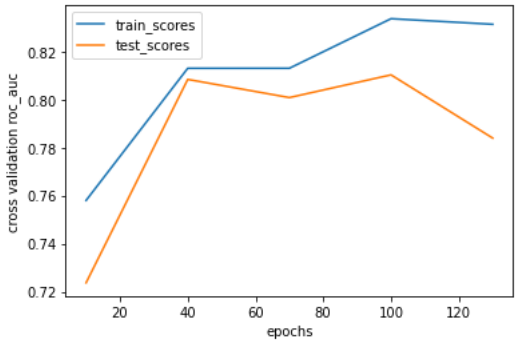


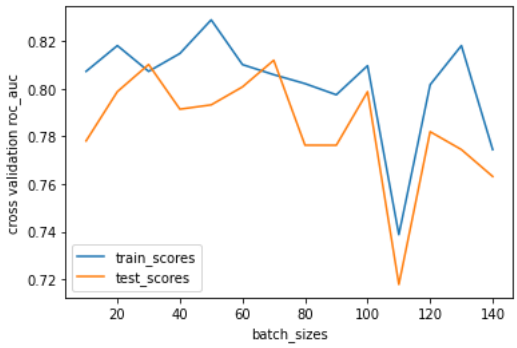


KNN is distance-based model. It classify the data according to the similarity(distance). More neighbors help the model improve roc auc but it also make the computation more complex. In the left grapby, I can see when n_neighbors is around 3, the testing score is high. After that, testing roc auc increases very slowly. So I prefer to choose n_neighbors=3. In the second graph, weights based on distance and equal weights make no difference on model performance. The differences on the training roc auc is just 0.06 which is very small.

5. Neural network



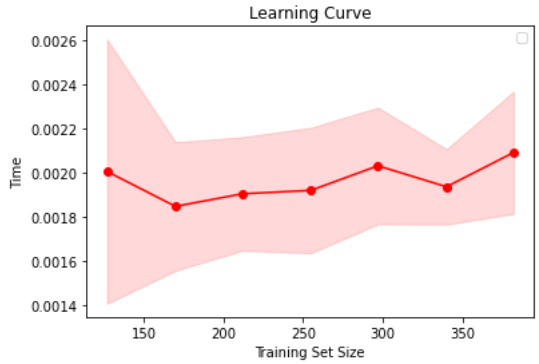


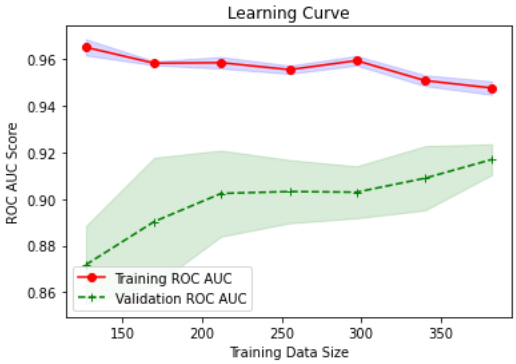


Overall, neural network performance is very stable. I set 1 hidden layer with 16 nodes using relu activation function and 1 outlay with sigmoid function. According to the graphs above, I prefer to choose 'Adam' that has high roc auc score. Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. For epochs, usually larger epochs will result in better model. But after epochs=40, the gap between training and testing become larger so epochs=40 is the best. For batch size, it look like smaller batch size is better because smaller batch size not only has faster training dynamics but also does better generalization to the test dataset versus larger batch sizes. In the last graph, I think batch_sizes=20 should work well.

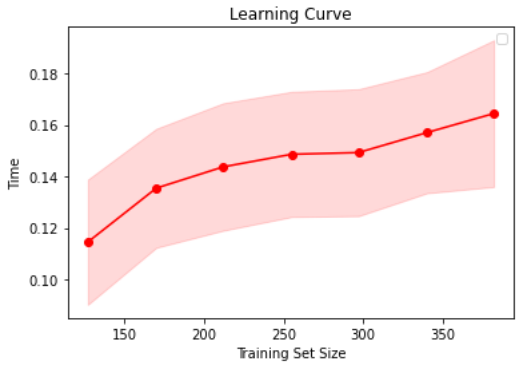
Learning Curve Analysis

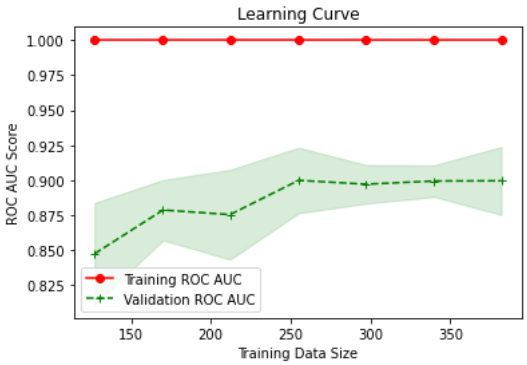
Decision Tree Learning Curve:



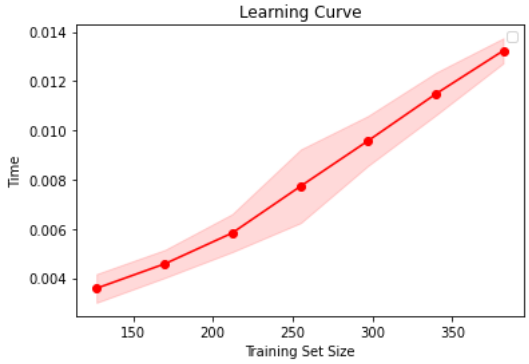


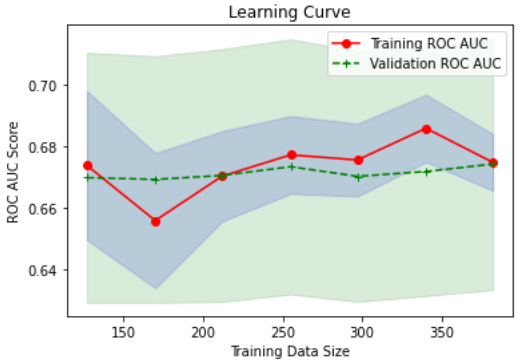
Gradient Boosting Learning Curve:



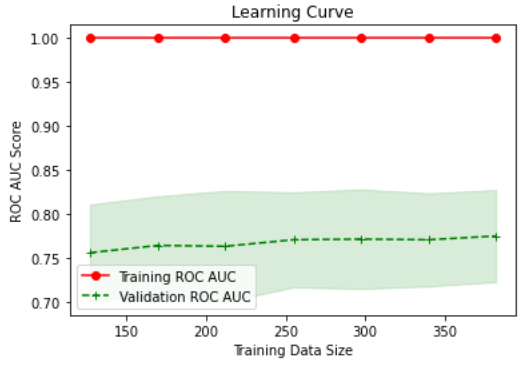


SVM Learning Curve Analysis:

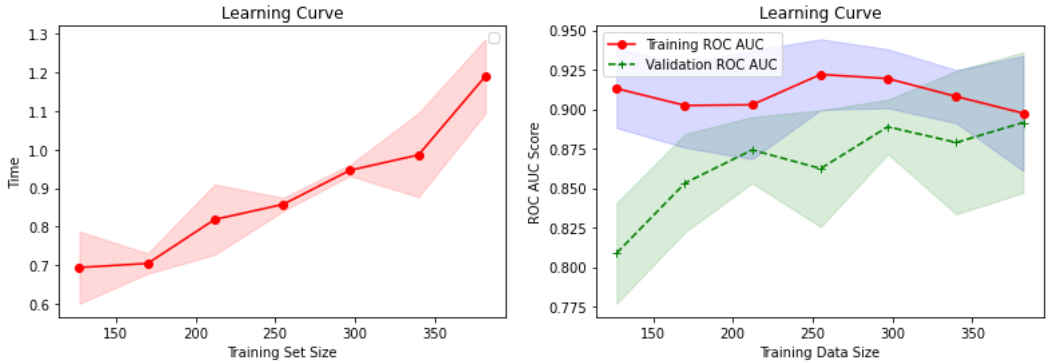




KNN Learning Curve Analysis:



Neural network Learning Curve Analysis



Comparing 5 model's learning curves on "time VS training size", I can see decision tree does not fluctuate a lot when training size increases. This data only has 690 rows. If we have more data, I expect the time will go up when training set size increases. KNN does not need training time and it only cares about K. KNN is a lazy learner. On the other hand, gradient boosting, SVM and Neural Network model training time goes up obviously. For gradient boosting model, more computation is needed when boosting method is applied on more training data since it focus on correcting the error in previous iteration. For SVM, it is a distance-based model so it also needs more time to find the best hyperplane when we have more data. For neural network, it will update the weights after each iteration. When we have more training data, it needs more time to compute the error and then do the backpropagation to update the weights.

Comparing 5 models' learning curves on "ROC AUC score VS training size", I can see decision tree and gradient boosting model have low bias and high variance, overfitting the data. More training instances adding to training set will help 2 lines converge. We can also add regularization or reduce the number of features to reduce the variance. For SVM, I can see training and validation roc auc scores are crossed each other and their performance are stable. This model has low variance. Since average roc auc is around 0.67, I suggest that we can try to get more features to train the model to improve roc auc. For KNN model, has low bias since training scores are very high overtime. But validation scores are also very low, and the gap is always large. So I think KNN has high variance since the data size is very small. Getting more data or using use regularization can be helpful to reduce variance. For neural network model, I can see training scores and testing scores converge, meaning they have low variance. In the meanwhile, both have high roc auc scores of 0.9. So this model performs very well since it has low bias and low variance.

Compare the models:

I tried the different combination to find the best model using 5 algorithms. The model performance is listed here including F1 score, accuracy, and ROC AUC score:

Models	F1	Accuracy	ROC AUC Score
Decision Tree	0.82	0.84	0.92
Gradient Boosting	0.76	0.78	0.89
SVM	0.84	0.87	0.90
KNN	0.82	0.84	0.91
Neural Network	0.74	0.79	0.81

In conclusion, I think decision tree works best because it has highest ROC AUC score. ROC AUC measures the quality of the model's predictions irrespective of what classification threshold is chosen. Decision tree's F1 score is also high. F1 score is the harmonic mean of the precision and recall. It also has high accuracy rate of 0.84. I believe if we get more data, decision tree model's variance will be reduced.

Dataset: Breast Cancer

Data (Breast_cancer_ta.csv) can be found in the link: https://www.kaggle.com/merishnasuwal/breast-cancer-prediction-dataset

Description of Classification problems:

This data set is used to determine whether the suspicious lump can cause breast cancer. We have 5 features to describe the lump: mean_radius, mean_texture, mean_perimeter, mean_area and mean_smooth. The target variable is diagnosis. 1 means cancer and 0 means non-cancer.

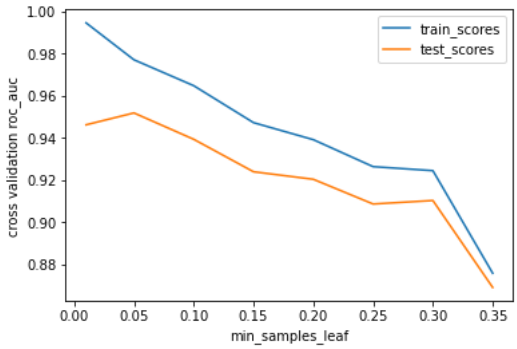
Why data is interesting:

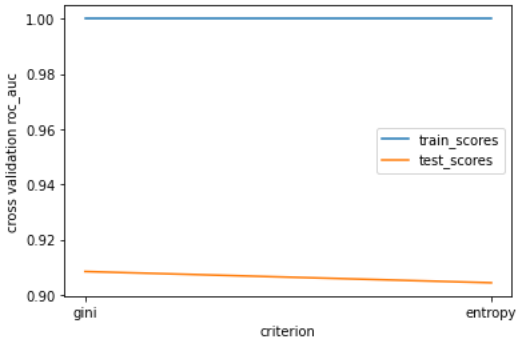
Worldwide, breast cancer is the most common type of cancer in women. Diagnosis of breast cancer is performed when an abnormal lump is found, or a tiny speck of calcium is seen. After a suspicious lump is found, the doctor will conduct a diagnosis to determine whether it is cancerous and, if so, whether it has spread to other parts of the body. So if supervised learning can be used to train a model to accurately classify the suspencious lump as the cause of cancer, patients would be beneficial from the model.

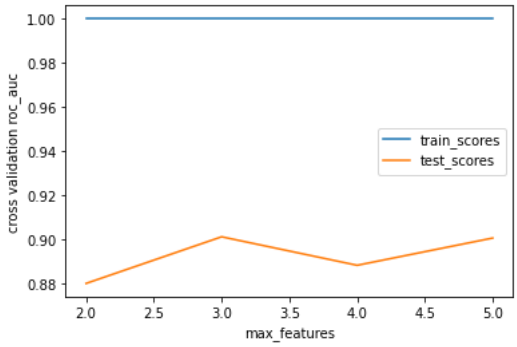
Parameter Tuning Analysis:

I applied 5 algorithms on the dataset. For each algorithm, I tried to tune parameters and see how the model performance changes. Then I use cross validation to select best combination of parameters which contribute to best model. Based on the best model, I plot the learning curve as a function of training data set to do analysis.

1. **Decision tree**



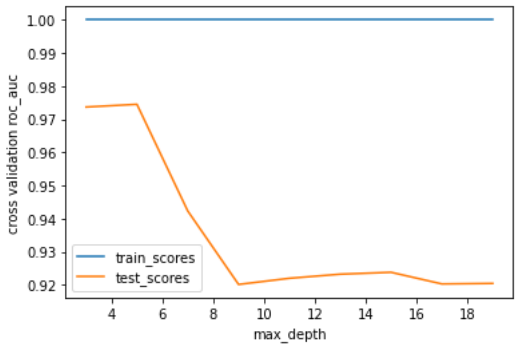


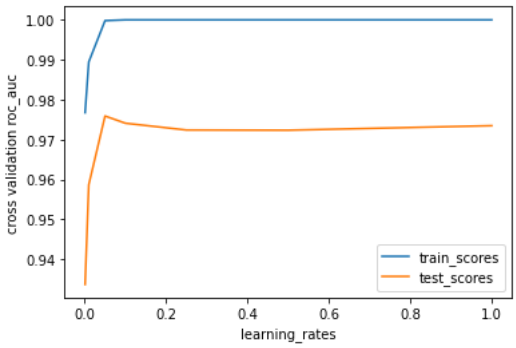


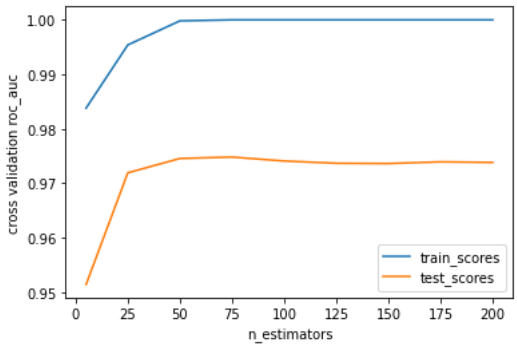
According to the graphs above, I can see when min_samples_leaf=0.05, the gap between training and testing lines are close and testing score is the highest. In the second graph, the performance on “gini” and “entropy” are almost same so splitting criterion using gini or entropy would not make any difference. For max_features, training scores do not change a lot. Testing scores

fluctuate a little bit but it ranges from 0.88 and 0.90. After max_features=3, the testing scores almost stay at 0.9, so I think max_features=3 is good enough because it makes model simpler.

2. Gradient Boosting

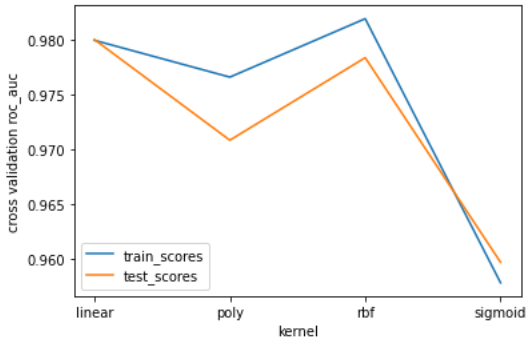


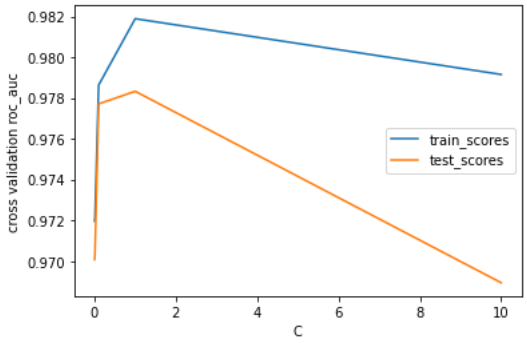




According to the graphs above. After max_depth=5, testing scores decrease quickly and training scores stay very high. So deeper trees will have low bias and high variance. That's why I suggest max_depth=5 which will avoid overfitting issue. For the learning rate, I think 0.1 is the best choice because after that, the training and testing scores do not increase and the gap stay same. For n_estimators, I suggest to choose 50 because when n_estimators=50, training and testing scores hit he highest point and will not increase after n_estimators=50.

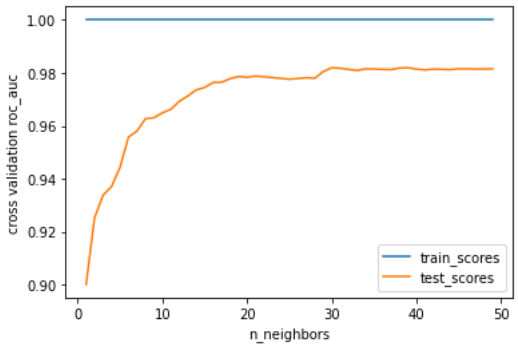
3. SVM

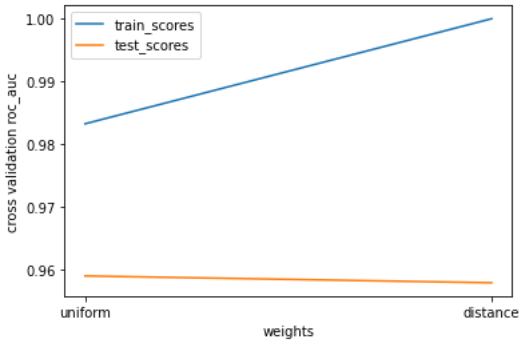




According to the graphs above, I think linear kernel is the best because training score and testing score are crossed each other. Both have high roc auc of 0.98. RBF kernel is the second best. Even though RBF kernel's training score is higher than linear kernel, its testing score is lower than training score so the model with RBF kernel has higher variance than the one with linear kernel. For the C, I think C=1 is the best because after that point, the training and testing scores go down and the gap between them become larger. When C=1, model has highest training and testing roc auc.

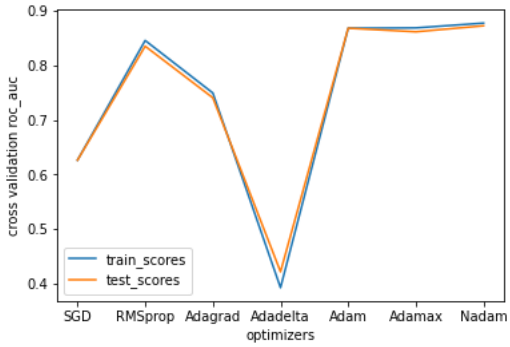
4. KNN

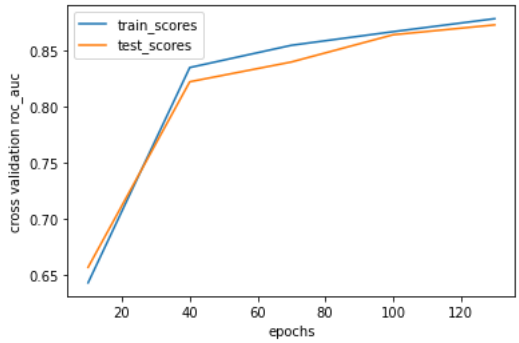


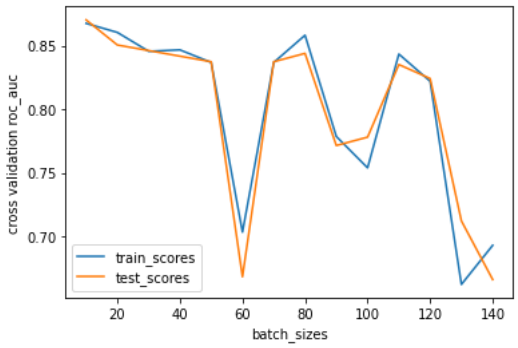


According to the graphs above, after n_neighbors=20, testing scores do not increase a lot. Compared to larger n_neighbors, n_neighbors=20 will make the model have relatively high roc and auc and reduce the computation work. For the weights, I think equal weights (uniform weights) and distance-based weights have similar performance. Just looking at the graph, people may think distance-based weights has higher training roc auc but the differnce is 0.02 which is very small. So I think it is because the dataset is small, choosing either of 2 types of weights will not affect model performance.

5. Neural network:



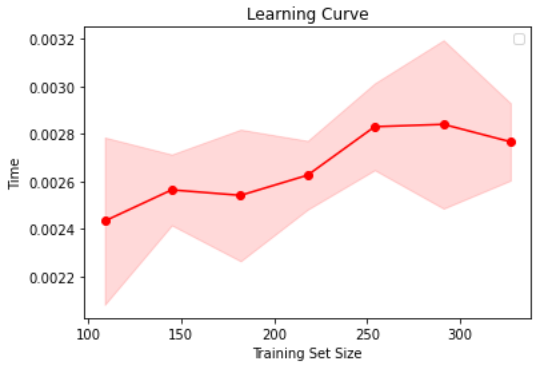


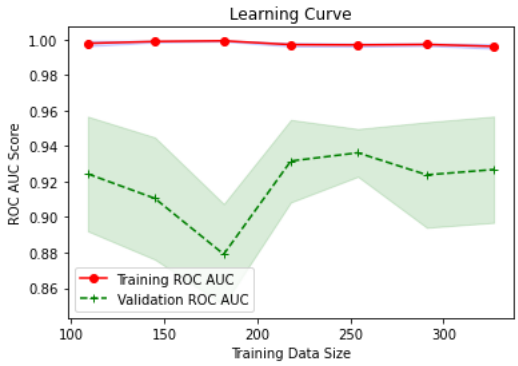


According to the graphs above, Adam, Adamax and Nadam are good optimizers we can use in the neural network model since their training and testing roc auc scores are high and similar. For the epochs, Training and testing scores have positive relationship with number of epochs. More epochs need more time to train the model. When epoch=40, the training and testing scores are close to each other and the scores are around 0.82. After epoch=40, both training and testing scores increase slowly. In the last graph, smaller batch size works better. Batch_sizes=10 has the highest roc auc score and training score and testing score are almost same.

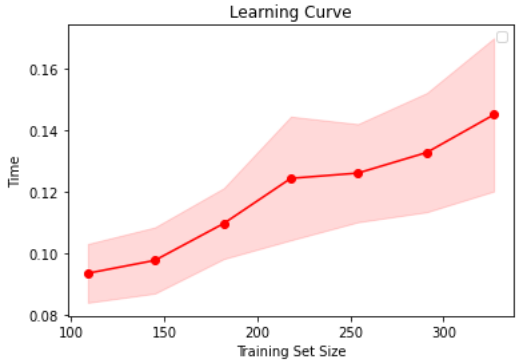
Learning Curve Analysis:

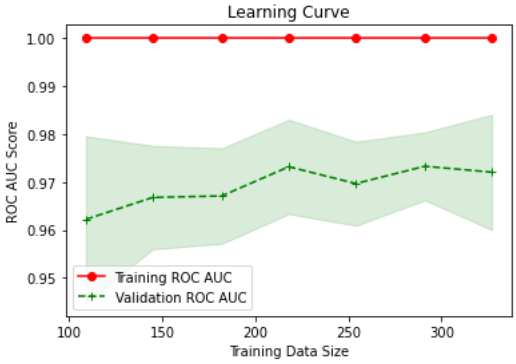
Decision Tree Learning Curve:



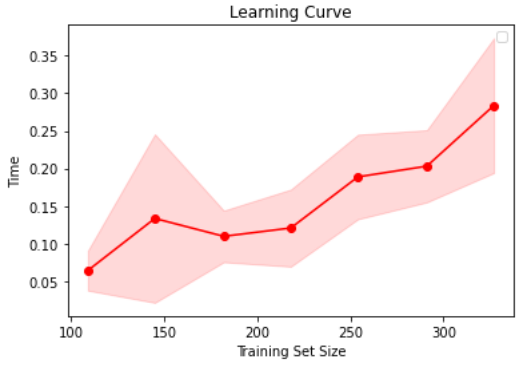


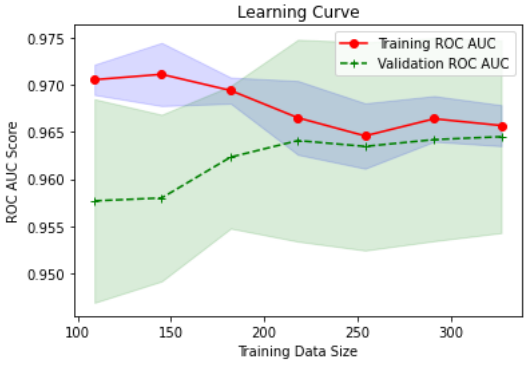
Gradient Boosting Learning Curve:



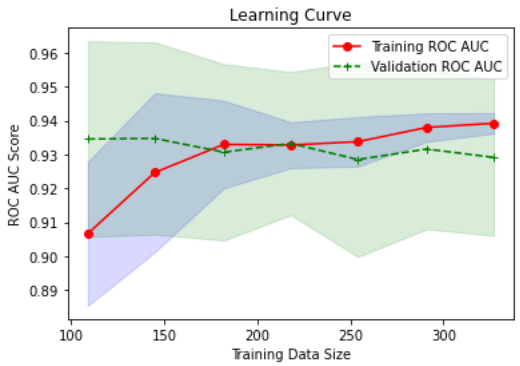


SVM Learning Curve:

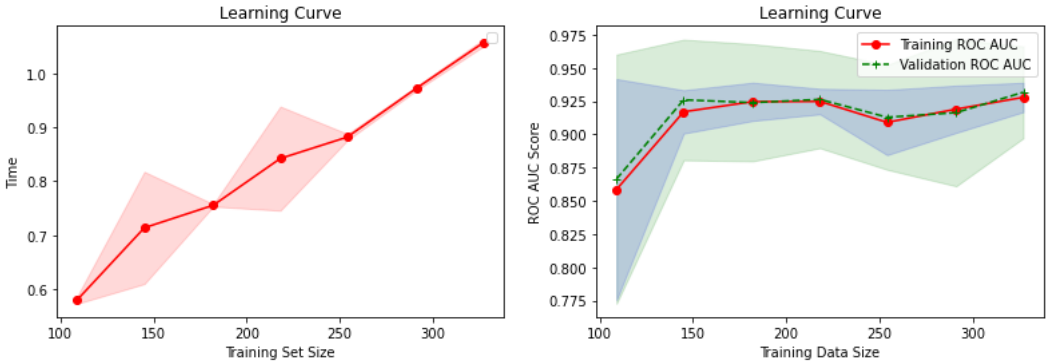




KNN Learning Curve:



Neural network learning curve:



Comparing 5 models' the learning curves on "training time VS training set size", I can see decision tree, gradient boosting, SVM and neural network models' training time increase because more computation is needed when data size goes up. Decision tree needs to compute the information gain when splitting the data. Gradient boosting needs to add more weights on the incorrectly predicted points in the previous iteration and then spend time focusing on correcting them in current iteration. SVM needs to find out the support vectors and the best hyperplane to maximize the margin. Neural network needs to do the back propagation to updates the weights.

Comparing 5 models' the learning curves on "training time VS training set size", I can see decision tree and gradient boosting have low bias and high variance because training scores are very high and validation scores are lower than training scores. Even though the gap between training and validation scores is large, The gap also becomes smaller when training size increases. If more instances are added, these 2 score lines will converge. SVM and neural network model have low variance and low bias. The training scores and validation scores converge together. Their roc auc scores are above 0.9 which is pretty good.

Compare the models:

I tried the different combination to find the best model using 5 algorithms. The model performance is listed here including F1 score, accuracy, and ROC AUC score:

Models	F1	Accuracy	ROC AUC Score
Decision Tree	0.94	0.92	0.94
Gradient Boosting	0.95	0.93	0.98
SVM	0.95	0.93	0.99
KNN	0.95	0.93	0.98
Neural Network	0.91	0.88	0.91

In conclusion, I think SVM works best. SVM model has low bias and low variance. This model's ROC AUC, accuracy rate and F1 score are very high. So it outperform than other models.