Name: Heng Chang Rong Kelvin
Course: CS7641
Assignment: 1

Supervised Learning Analysis

Introduction

The goal of this assignment is to compare five different learning algorithms (Decision tree with pruning, Boosting, K-nearest neighbors, support vector machines and neural networks) on two different interesting datasets (Steel pipe dataset and Contraceptive method dataset).

Dataset

The Steel Plates dataset is interesting because accurate predictions on it can help to automate the process of diagnosing and changing steel plates when it is too dirty, allowing us to have a wide use of applications in many industries.

The contraceptives dataset helps to determine whether contraceptives are being used based on many attributes of the husband and wife. Modelling birth rates, sales of contraceptives and healthcare related applications are some of the interesting use cases from solving this dataset

Preprocessing

For both datasets, we keep the preprocessing to be the same across all algorithms. The steps for preprocessing are shown below:

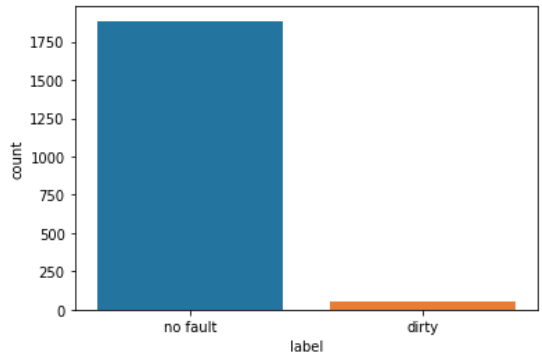
Steel plates

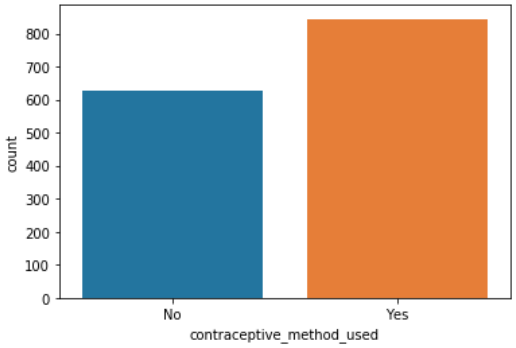
1) Keep only the dirtiness label
2) Scale the attributes of the steel plates to between 0 and 1
3) Split out a test set from the original data using stratified shuffle split

Contraceptive choice

1) Clean the label to include only ‘no contraceptives used’ and ‘contraceptives used’
2) Use one hot encoding to encode the categorical variables
3) Since the numerical values are all roughly within the same order of magnitude, we do not scale them to between 0 and 1
4) Split out a test set from the original data of about 25% using stratified shuffle split

Distribution of class labels





We can see from the distribution charts above that the class labels is very imbalanced (2.83% class 1) in the steel plates dataset as compared to the contraceptives dataset (57.2% class 1)

Methodology of comparison

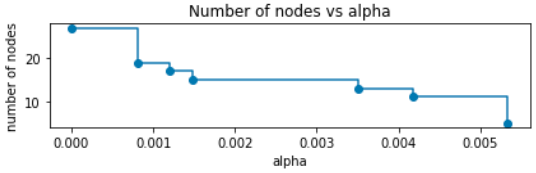
In all our comparison under this section, we use the sklearn library and the following framework for comparison:

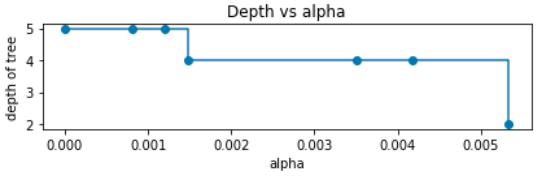
Since we have an imbalanced dataset, area under the precision recall curve (AUPRC) is chosen as our metric to penalize mistakes on minority class more heavily. We also try to tune the hyperparameters using 5-fold cross validation and predict on the test set to get the optimal test set AUPRC. We fix the optimal hyperparameters and conduct a learning curve analysis to understand the bias/variance tradeoff. Next, we conduct model complexity analysis by varying 2 different parameters of each algorithm. Lastly, we also obtain the time taken to train and predict for each algorithm.

Steel Plates Dataset

Decision Tree

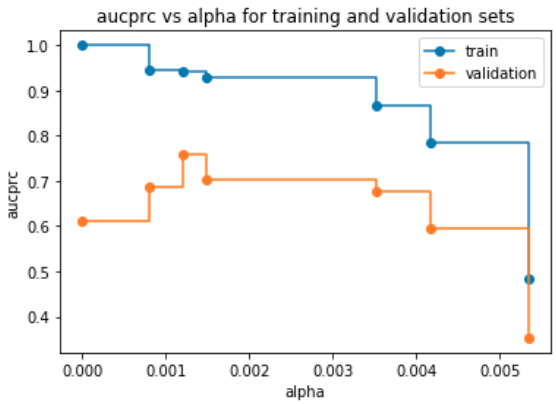
In order to avoid severe overfitting for decision tree, we use pruning and investigate the effect of alpha on the number of nodes and depth of the tree.

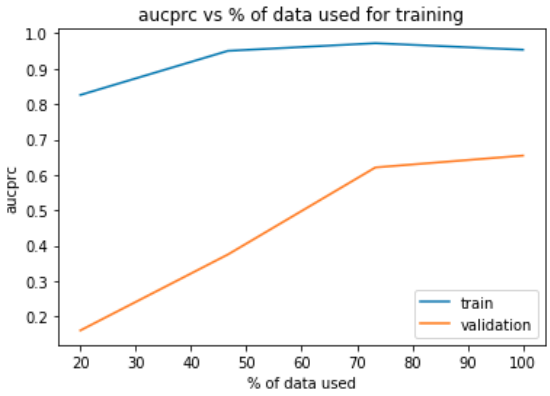




From the diagram above, it is clear that by increasing the alpha value, we can reduce the number of nodes and depth of the tree to avoid overfitting on the training data.

Next, we then analyse the train and validation AUPRC by varying the pruning parameter alpha on the train set.



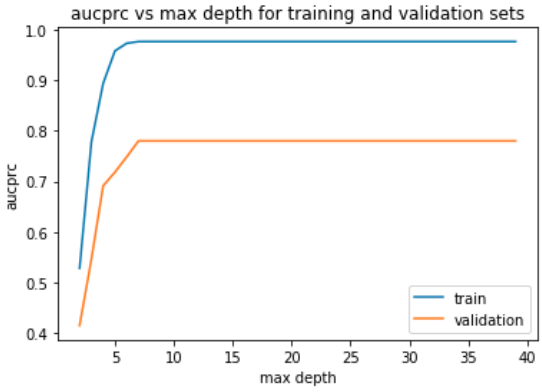


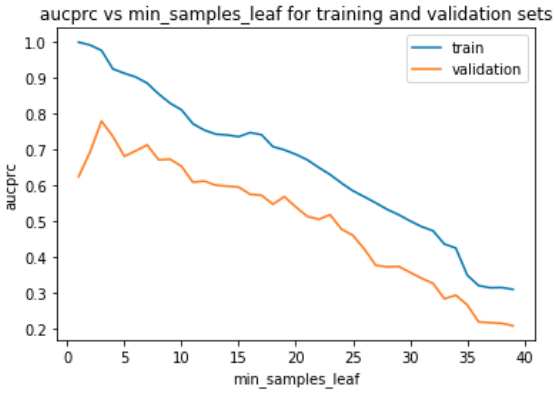
We can see that the train AUPRC will decrease monotonically as we increase the level of pruning to avoid overfitting. However, we can see that the validation AUPRC reaches an optimal level at around alpha = 0.0012 with the generalisation of the tree.

Next, we conduct a hyperparameter search of max_depth, min_samples_leaf and ccp_alpha to obtain optimal parameters {'ccp_alpha': 0.000261, 'max_depth': 43, 'min_samples_leaf': 3}. With this set of parameters, we conduct the learning curve analysis below:

It is evident that the increase in amount of data used for training helps to boost the validation AUPRC. Furthermore, there is a large gap between both curves, which allows me to conclude that the model may have overfitted a little.

Lastly, we varied two different parameters of the tree while fixing the rest of the optimal parameters the same and obtained the following charts:

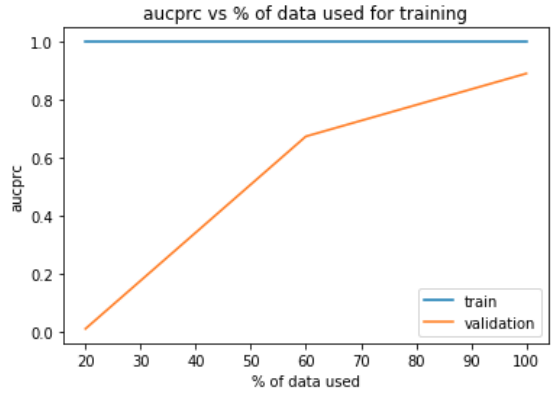


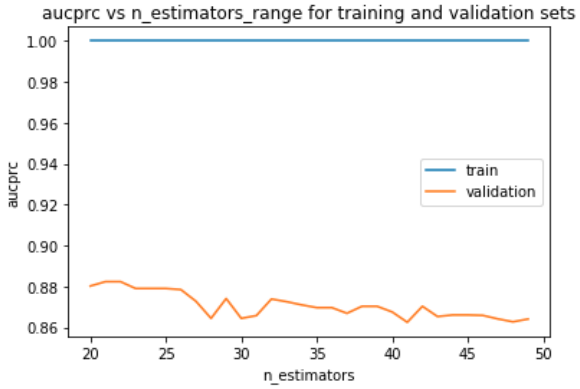


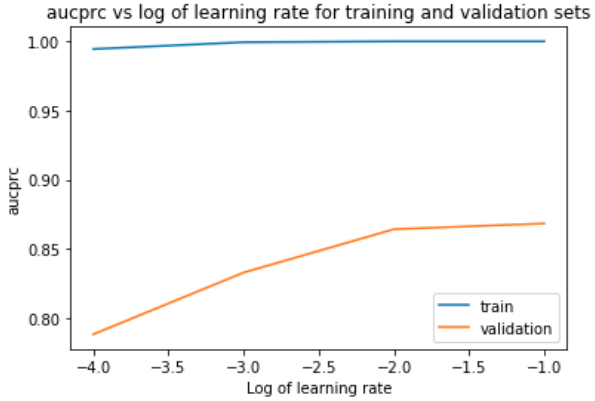
It is clear that the AUPRC plateaus after increasing max depth beyond 10 since we apply pruning. Finally, we also see that train AUPRC drops monotonically as we increase minimum samples in each leaf to prevent overfitting, whereas validation AUPRC peaks at a value of 3.

Boosting

The boosting method chosen is Adaboost, with the optimal decision tree in the previous section as the underlying base estimator. Following the same methodology described earlier, we obtained the following set of optimal parameters {'learning_rate': 0.01, 'n_estimators': 21} and the graphs below:



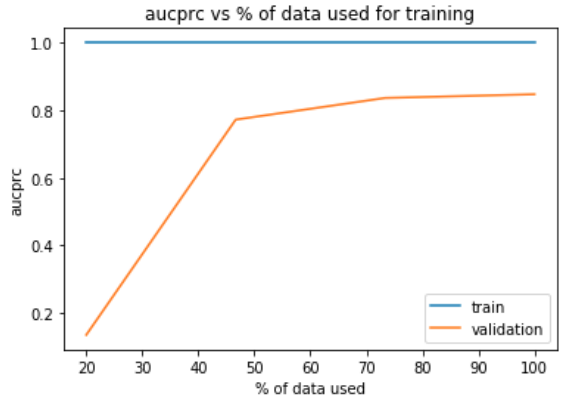


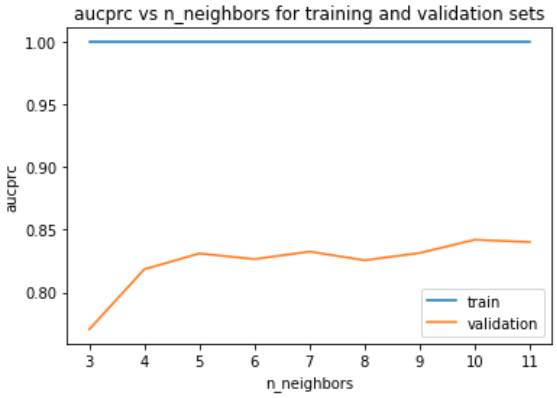


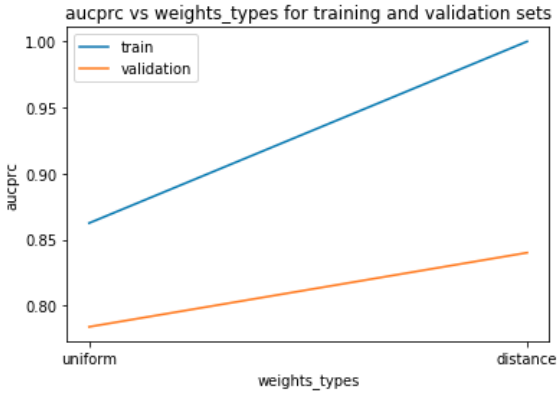
It is evident that increasing the amount of training data did help the validation AUPRC. We also observed that the train AUPRC is almost perfect for all graphs. This is because boosting algorithms seeks to reduce error on each iteration, and may have resulted in overfitting. Furthermore, we also see that there is an optimal number of estimators and learning rate. Too few or many iterations will result in underfitting and overfitting respectively. Similarly, we need an optimal learning rate to speed up the rate of learning and not to fall into local optimum.

K-Nearest Neighbors (K-NN)

Next, we built a K-NN model on the data. Following the same methodology described earlier, we obtained the following set of optimal parameters {'leaf_size': 29, 'n_neighbors': 10, 'p': 1, 'weights': 'distance'} and the graphs below:



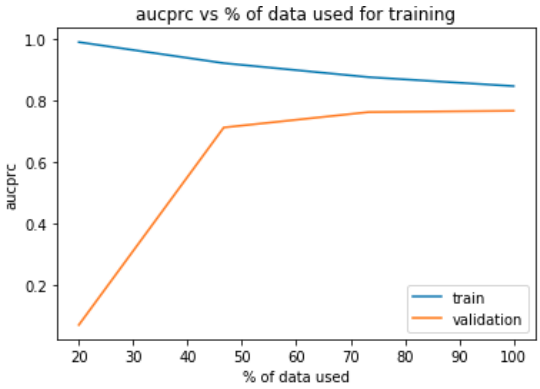


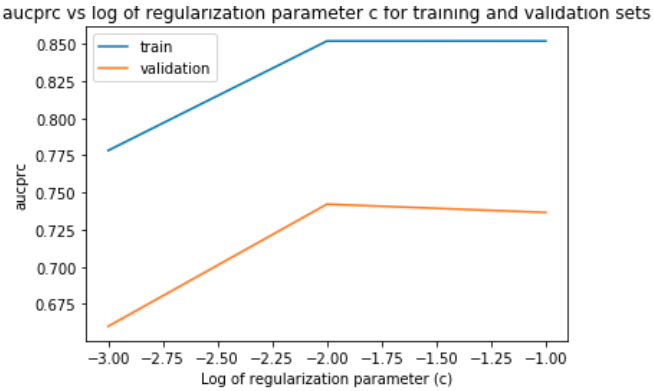


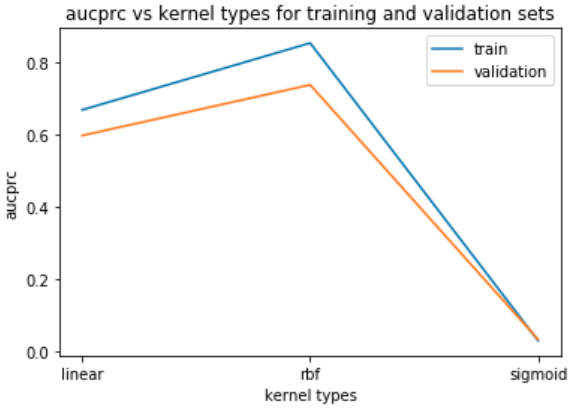
It is evident that increasing the amount of training data did help the validation AUPRC, but the effect plateaus off after 40% of the training data. We notice that the train AUPRC is near 1 for the various graphs with ‘distance’ weight types. The reason seems to be because we can find an optimal number of neighbors which fits the training data perfectly. Furthermore, we also see that there is an optimal number of nearest neighbors and weight types. The optimal number of nearest neighbors seems to be 10, i.e prediction of any data point depends on its 10 nearest neighbors. Also, having ‘distance’ as the weight type seems to be better than ‘uniform’, i.e any data point with more similar distance in terms of attributes has a greater influence, which is intuitively true.

Support Vector Machines (SVM)

Next, we fit an SVM on the same dataset and tuned the hyperparameters to arrive at the set of optimal parameters {'C': 0.07472491075598509, 'coef0': 3.9626217315659873, 'gamma': 'scale', 'kernel': 'rbf', 'shrinking': False}, learning curve and model complexity analysis below:



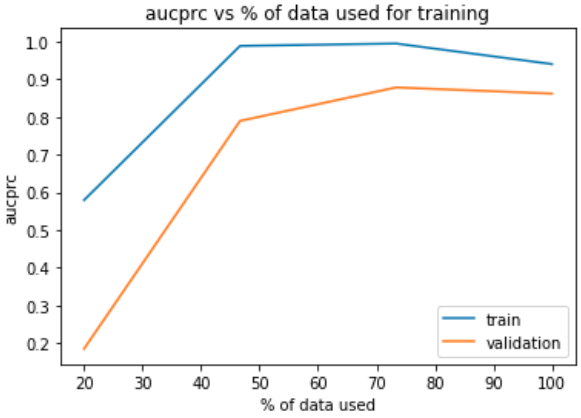


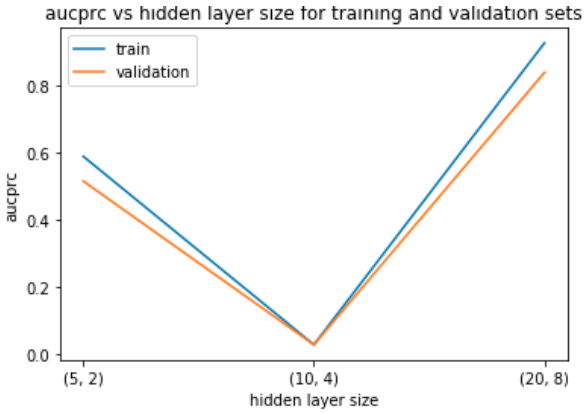


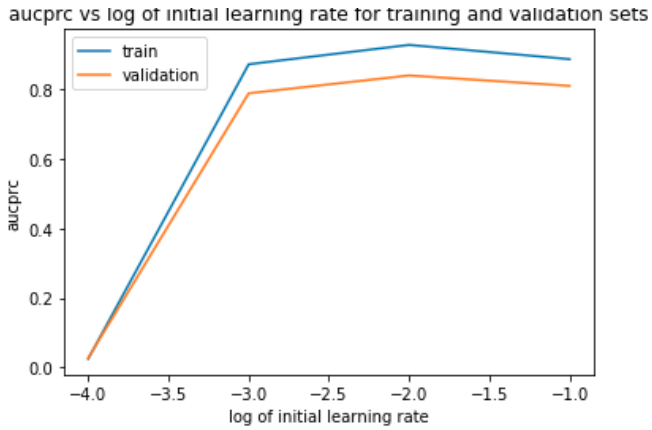
Similar to the learners above, the validation AUPRC increases monotonically and plateaus off after 40% of the training data used. The model complexity analysis also shows that regularization parameter = 0.01 and kernel type = radial basis function (rbf) seems to be working the best. Intuitively, too much regularization actually hurts the model performance as it will not be able to model complicated relationships between variables, whereas little to no regularization leads to overfitting. However, this relationship is not shown in the graphs presumably because of the shuffling of small dataset. The rbf kernel is able to do well because of its capability to discriminate between classes using squared euclidean distance between the feature vectors.

Neural Networks

Finally, we fit a shallow 2 layer neural network to the data tuned the hyperparameters to arrive at the set of optimal parameters {'early_stopping': False, 'hidden_layer_sizes': (20, 8), 'learning_rate_init': 0.01, 'max_iter': 2146, 'validation_fraction': 0.248}, learning curve and model complexity analysis below:

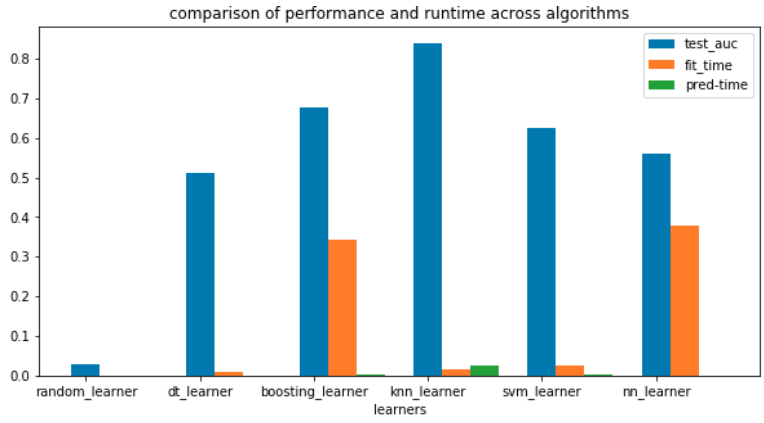






Similar to the learners above, the learning curve analysis shows that the validation AUPRC increases monotonically and plateaus off after 50% of the training data used. The model complexity analysis tells us that for 2 layer neural networks, 20 and 8 hidden units are the best combinations among the hyperparameters we tried. Small amount of hidden units may not be able to model complex relationships in the data. Finally, we also find that a learning rate of 0.01 seems to be the best. A lower learning rate may result in non-convergence after maximum number of iterations, whereas higher learning rate may not result in global optimum.

Comparison of algorithms

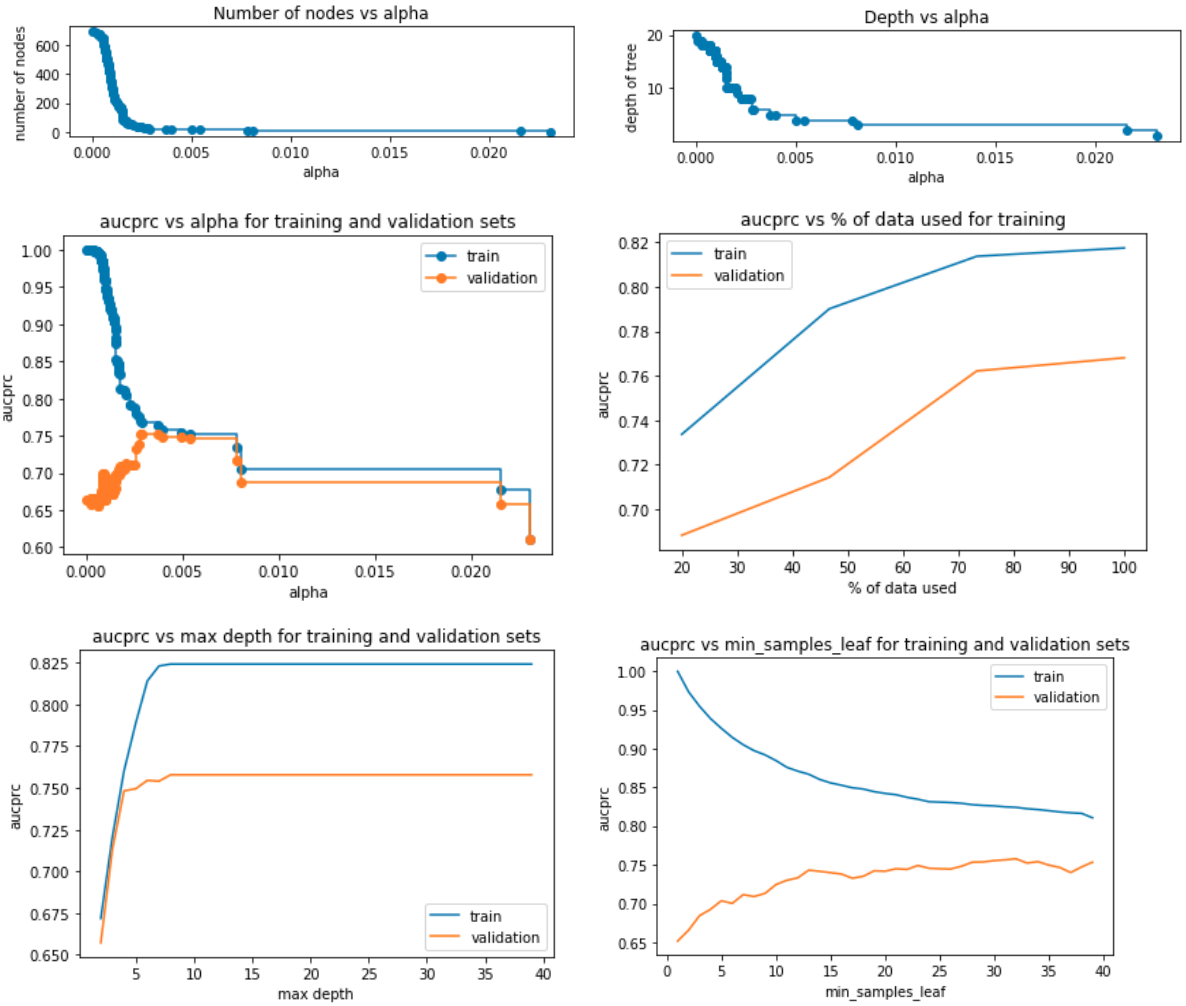


With the same preprocessing and analysis methods done across the five algorithms, we can compare their training time, prediction time and performance. We also added a random learner which just predicts class 0 or 1 with 50% probability each. From the charts above, we can see that the K-NN algorithms performs the best on the steel plates dataset with negligible training time. Although its prediction time is the highest, it is not unreasonably high because our dataset is sufficiently small. K-NN may perform the best because of the dataset is class imbalanced and also able to be separated reasonably well in high dimensional space. If the minority class is very close together in terms of its feature, the optimal hyperparameter of 10 nearest neighbors is sufficient to always predict with high confidence. It is also interesting to note that boosting is able to achieve a much higher validation AUPRC, but not test AUPRC perhaps due to overfitting. Given more time, I would have done a more comprehensive hyperparameters gridsearch.

Contraceptives dataset

Decision Tree

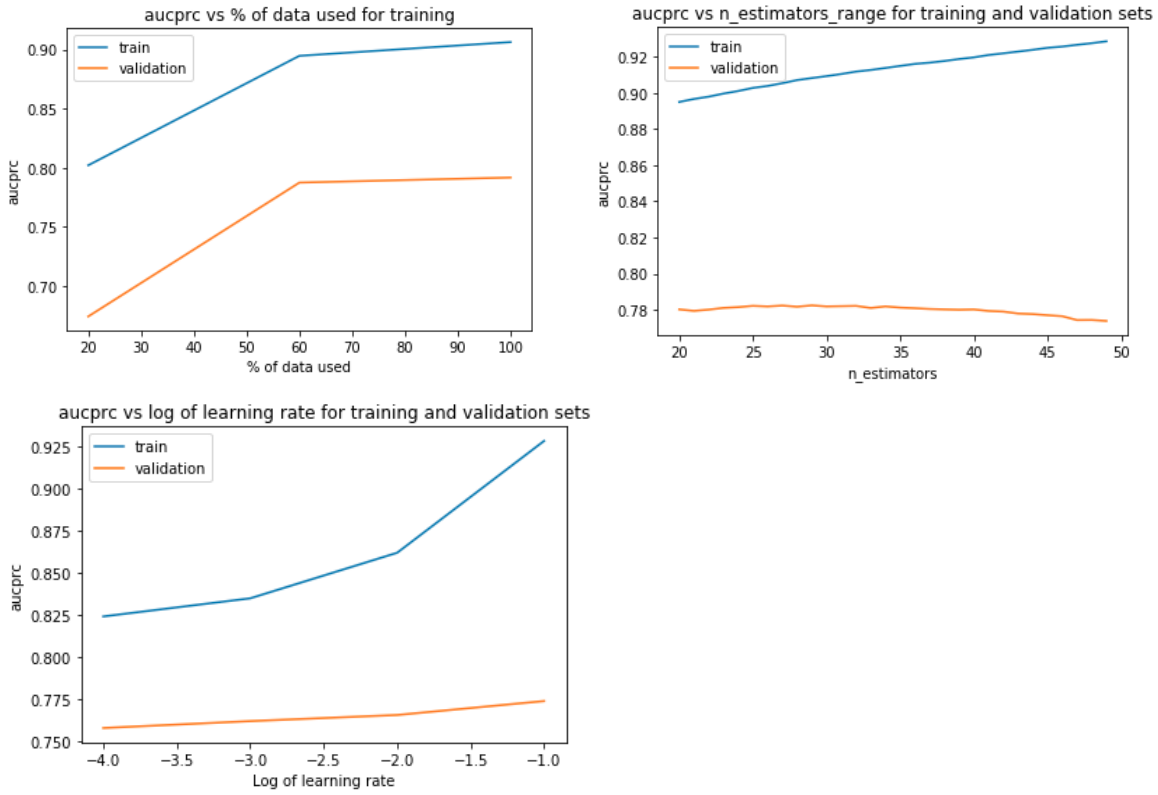
Similar to the analysis in previous dataset, we also investigated pruning, conducted learning curve analysis and model complexity analysis and obtained the following:



We also see a similar pattern in the pruning graphs. The train AUPRC decrease monotonically while validation AUPRC peaks at around alpha = 0.0025. A basic hyperparameter search also gave the parameters {'ccp_alpha': 0.0000449, 'max_depth': 29, 'min_samples_leaf': 32}. Similarly, we also see that more data helped with the validation AUPRC. Furthermore, the model complexity analysis also shows an optimal level of minimum samples of leaf and max depth (due to pruning).

Boosting

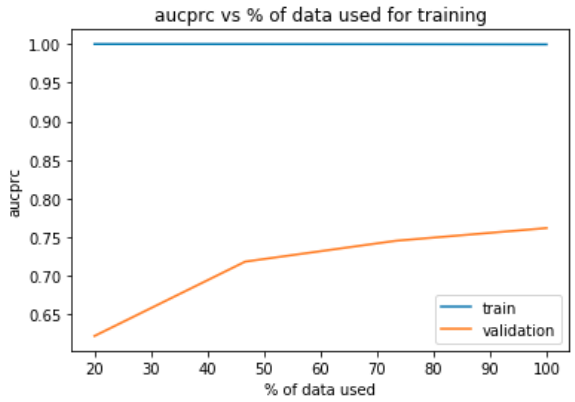
Similarly, we chose Adaboost with the above decision tree as the underlying base estimator. Following the same methodology described earlier, we obtained the following set of optimal parameters {'learning_rate': 0.1, 'n_estimators': 29} and the graphs below:

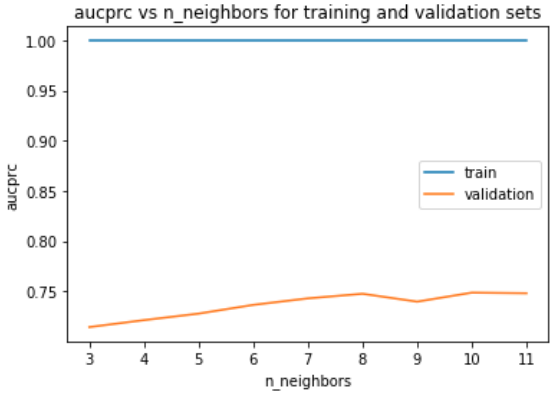


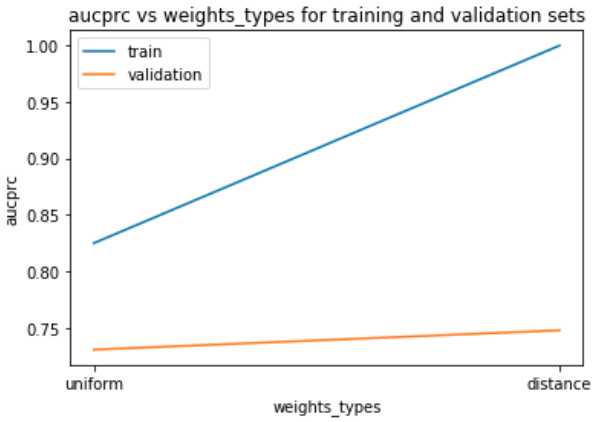
It is evident that increasing the amount of training data did help the validation AUPRC. However, the AUPRC did plateau off after 60% of the training data Furthermore, we also see that there is an optimal number of estimators and learning rate. The reasons for these are the same as those in the boosting section of steel plates dataset.

K-Nearest Neighbors (K-NN)

Similarly, following the same methodology described earlier, we obtained the following set of optimal parameters {'leaf_size': 10, 'n_neighbors': 10, 'p': 2, 'weights': 'distance'} and the graphs below:



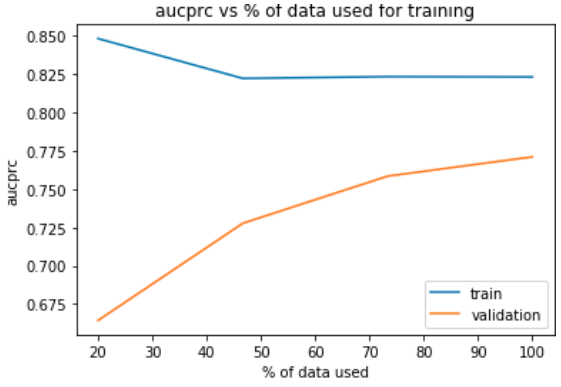


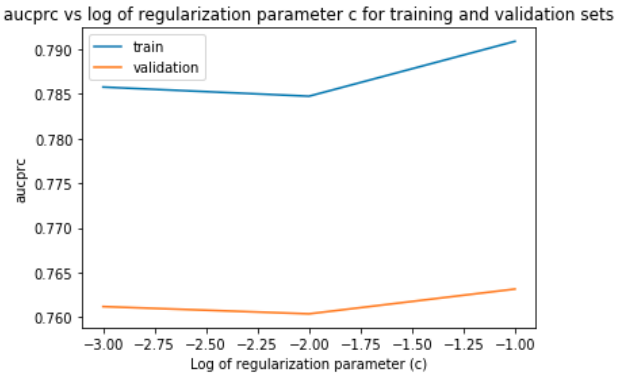


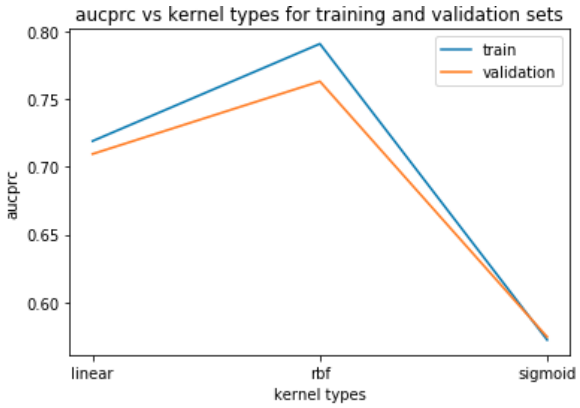
It is evident that increasing the amount of training data did help the validation AUPRC. Furthermore, we also see that there is an optimal number of nearest neighbors (10) and weight types (distance). The reasons for these are the same as those in the K-nearest Neighbors section of the steel plates dataset.

Support Vector Machines (SVM)

Next, we fit an SVM on the same dataset and tuned the hyperparameters to arrive at the set of optimal parameters {'C': 0.42026649471794364, 'coef0': 1.2582104044983022, 'gamma': 'auto', 'kernel': 'rbf', 'shrinking': True}, learning curve and model complexity analysis below:



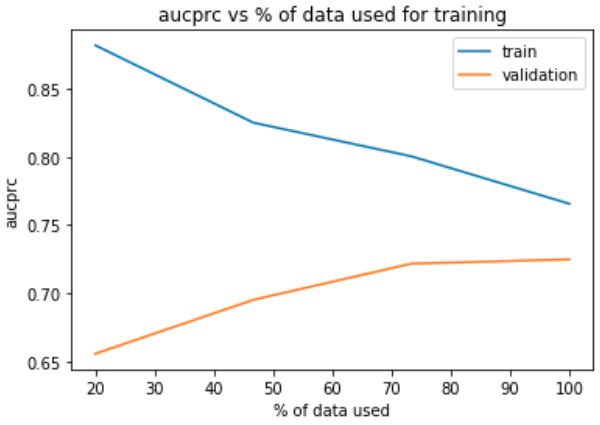


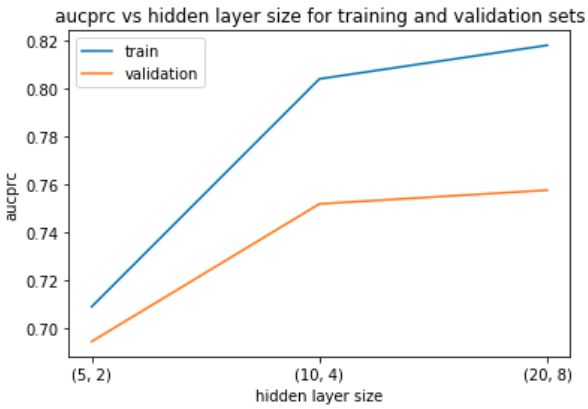


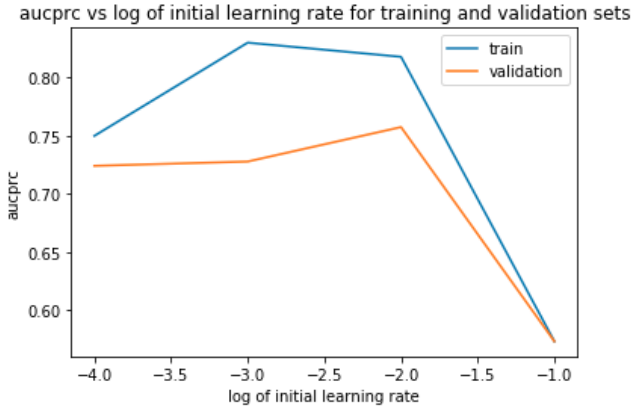
From these graphs, we see a very similar relationship compared to the previous steel plates dataset. Hence, the explanations are the same as above. The only difference we see is that we need a large regularization parameter of 0.1 to do well.

Neural Networks

Finally, we also fit a shallow 2 layer neural network to the data tuned the hyperparameters to arrive at the set of optimal parameters {'early_stopping': False, 'hidden_layer_sizes': (20, 8), 'learning_rate_init': 0.01, 'max_iter': 2524, 'validation_fraction': 0.263}, learning curve and model complexity analysis below:

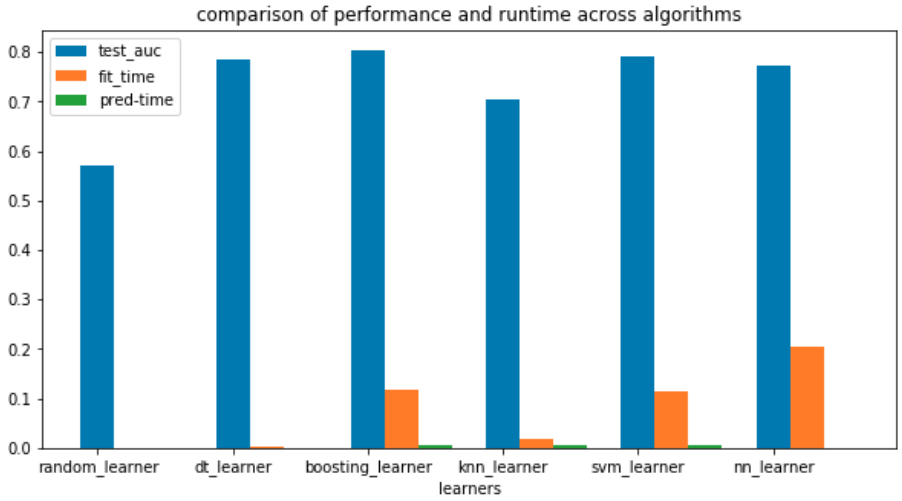






According to the learning curve, more data indeed help with the validation AUPRC. The model complexity analysis tells us that for 2 layer neural networks, 20 and 8 hidden units are the best combinations among the hyperparameters we tried. Finally, we also find that a learning rate of 0.01 seems to be the best. The explanations can be found in the Neural Networks section in the first dataset.

Comparison of algorithms



From the charts above, we can see that the Adaboost algorithm performs the best on the contraceptives dataset with moderate training time and negligible prediction time. I will consider boosting to be production friendly since the prediction time is short.

Conclusion

After a thorough analysis on five algorithms across two interesting datasets, I realize that machine learning is a Science and also an Art. The ‘Art’ part is that machine learning is about trying different datasets and experimenting what works well, whereas the ‘Science’ part is the underlying mathematical formulation behind each algorithm.

References

1. Steel Plates Faults Data Set – UCI Machine Learning Repository
https://archive.ics.uci.edu/ml/datasets/Steel+Plates+Faults

2. Contraceptive Method Choice Data Set - UCI Machine Learning Repository
https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice