Machine Learning Algorithms comparison and analysis

902905501
Yuanheng Wang

Problem Description:

Two datasets are selected for the purpose of this assignment. One is “Bank Marketing”, and the other is “Chess (King-Rook vs. King-Pawn)”. Both datasets are acquired from UCI Machine Learning Dataset Repository. Below are the explanations of why to choose these two datasets.

Bank Marketing:

The data is related to a marketing campaign of Portuguese banking institution. The data contains 20 attributes(and the final output result is whether people with certain attributes would choose to subscribe the service provided. The full dataset contains 45211 instances and is a typical classification problem related to business field. I found the problem extremely interesting because the concept behind this dataset can be expanded all sorts of areas in the real world, especially for business undergoing big decisions on whether or not to launch marketing campaigns and relative strategies. For instance, imagine Microsoft is evaluating and researching the market on deciding whether to launch a new bundle of Xbox with a third party bundle game. If information of customers can be retrieved from previous launch events, including their ages, occupations, estimated annual salaries and geographic information, Microsoft would be able to identify the most efficient marketing strategy and tailor the channels of marketing in different areas. The subject can be extended to relevant social studies as well. Government usually have the most thorough information of the residents. If local government wants to push a service or a new proposal, this similar methodology can be applied to give an approximate possibility of passing certain bills.

King-Rook vs. King-Pawn:

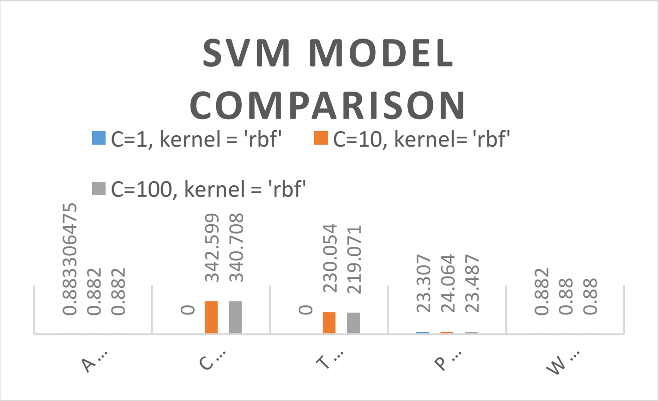
This is a typical game problem that I believe will introduce me to the concept of game AI. The situation is as follows: White has King and one Rook left on the chess board, Black has King and one Pawn left on the position of A7, which is one step away from advancement. The data contains 36 attributes, and the 37th is the result whether white can win or not. The part I find interesting about this dataset is its attributes. Each attribute is a state description and all those 36 attributes combined describe the entire chessboard. When the system becomes more complicated, for example, Starcraft, there will be more attributes describing the situation of the game. But in principle, I think the concept should be fairly similar. To be more general, not only games, but anything in real world that can be depicted by multiple attributes combined should allow us to apply same thought process.

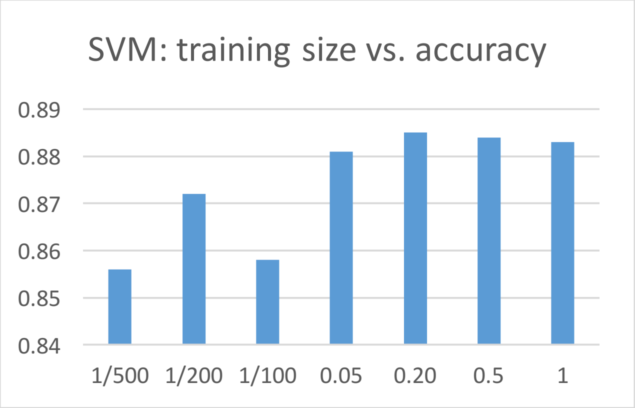
Algorithm Comparison:

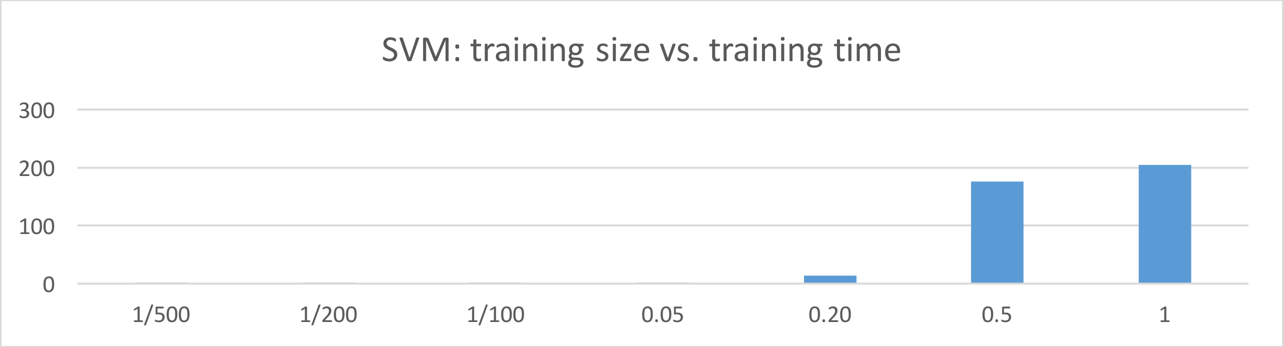
All the results below are attained by tuning the parameters, finding the best parameter settings for one algorithm. By splitting the dataset into training and real-world untouched data, I used cross validating method for the training dataset to get the best model. Then test the model with untouched real world model to get the most unbiased prediction accuracy.

Bank Marketing – SVM:

The SVC classifier in sklearn is implemented.
The tested parameters are
Parameter = {‘C’:[1,10], ‘kernel’: (‘linear’, ‘rbf’)}. One issue to be mentioned is for this dataset, the ‘linear’ kernel is extremely slow, and a result can only be obtained by shrinking the training size to 1/400 of the original size and skip the cross validation step. The accuracy is relatively lower than ‘rbf’ kernel with the same training size, but the training time required is much higher.

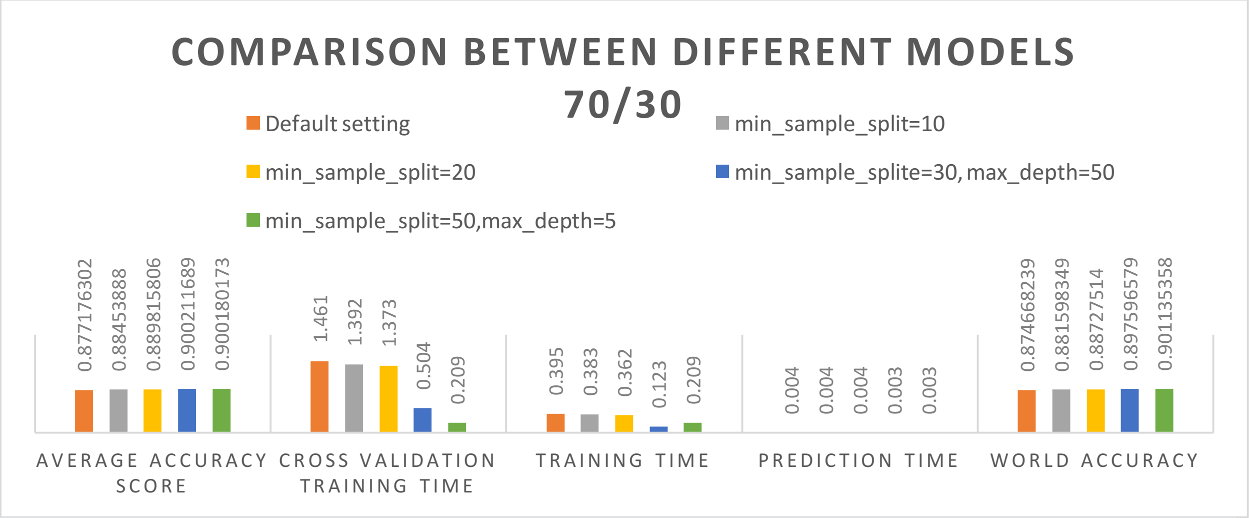


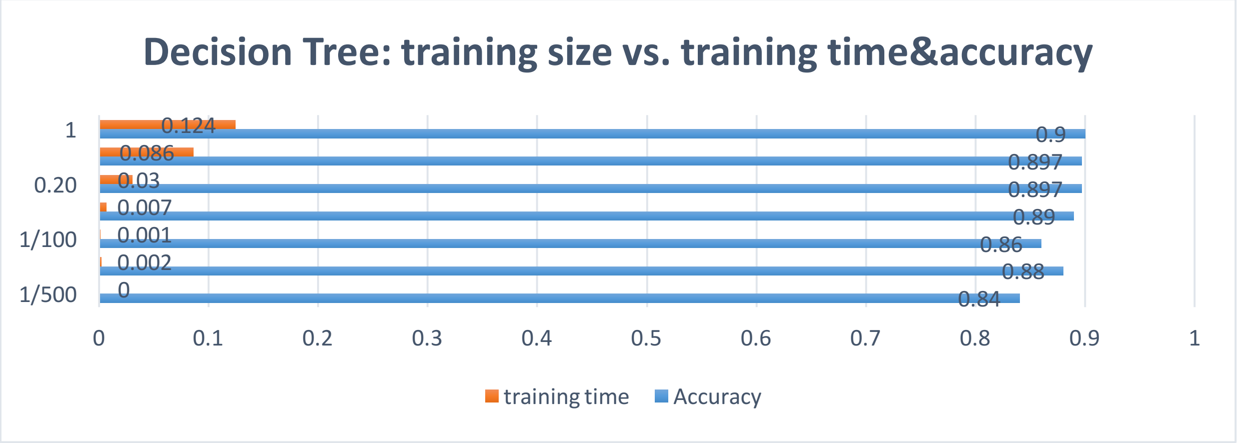




Bank Marketing – Decision Tree

The DecisionTreeClassifier in sklearn is implemented in this step. Below is the table of data.
A more direct graph representation.

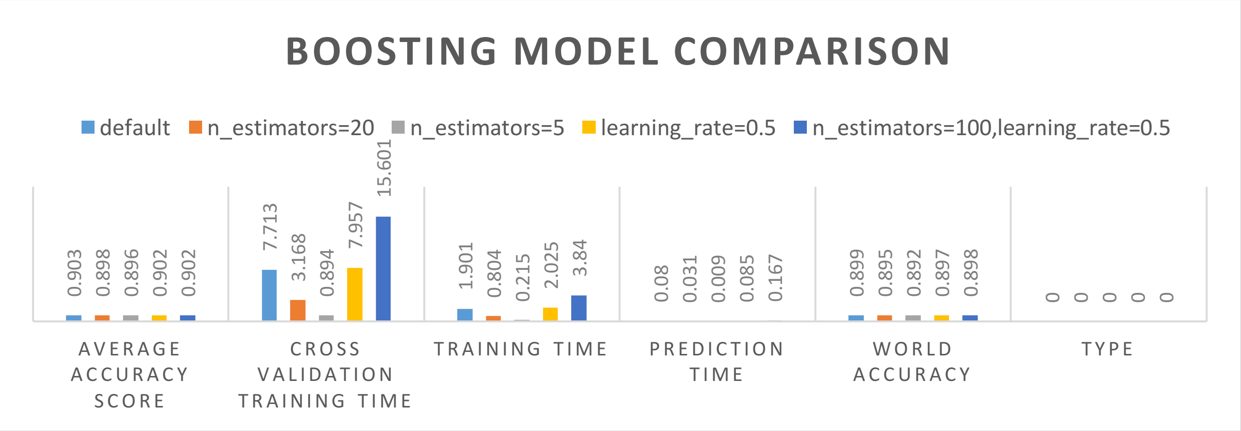


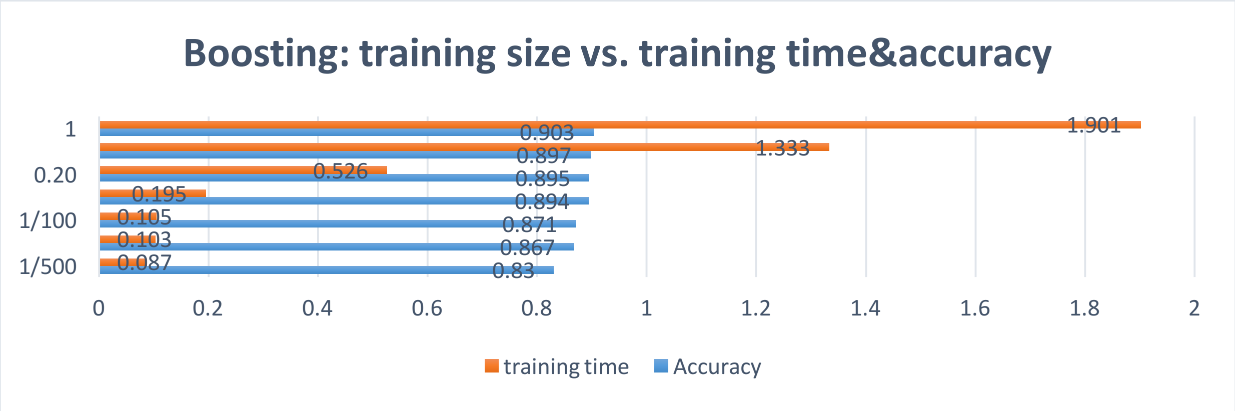


Decision Tree is doing decently well in this problem. The time it took to train was much faster than SVM, and the accuracy is nearly the same. The model with default setting actually took the longest time to do cross validation and came up with the worst accuracy. The cross validation time and training time were largely deducted when max_depth parameter was specified and with less training and prediction time, setting max_depth equal to 5 improves the accuracy. This indicates that by default decision tree tends to be overfitting with given dataset for this problem.

Bank Marketing – Boosting

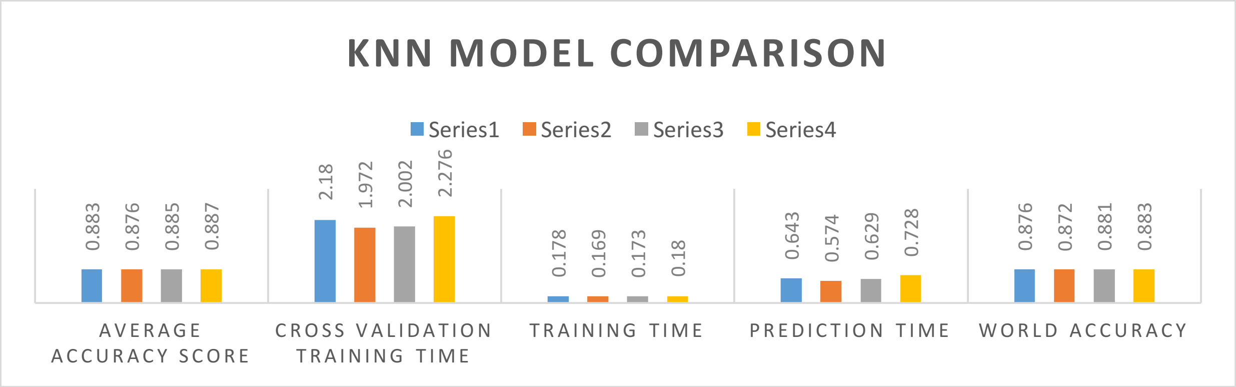
The AdaBoostClassifier in sklearn is implemented in this step. Below is the detail data.

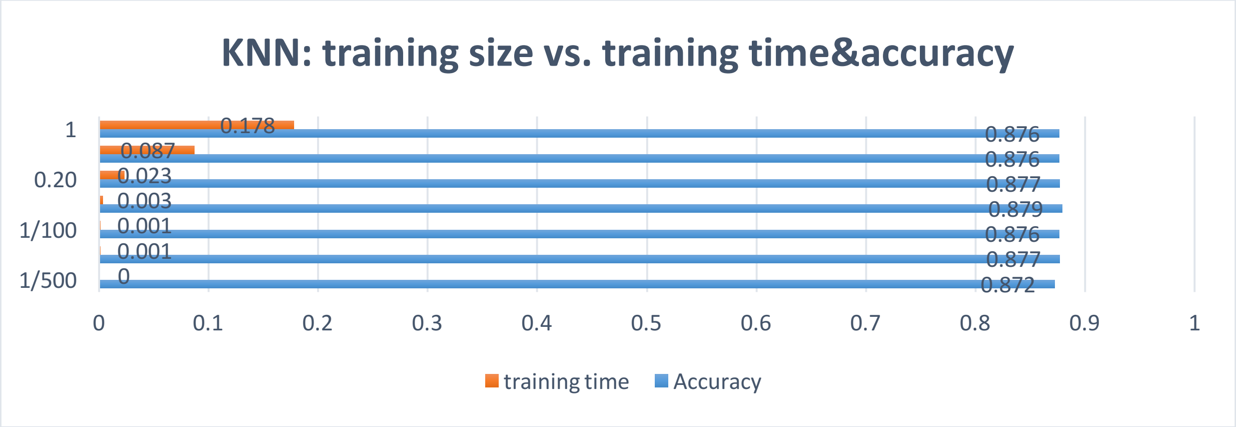




Bank Marketing – K-nearest Neighbors

The KNeighborsClassifier in sklearn is implemented in this step. Below is the detail data.

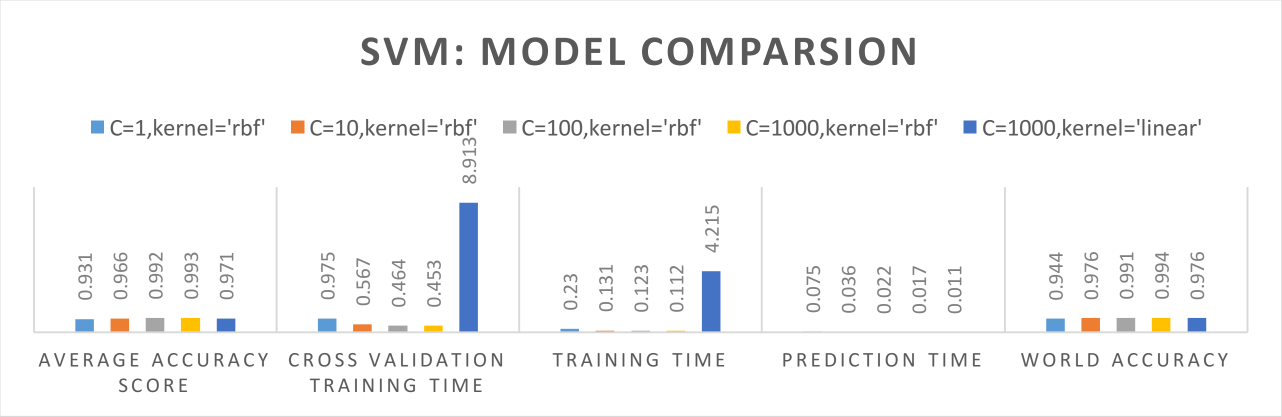


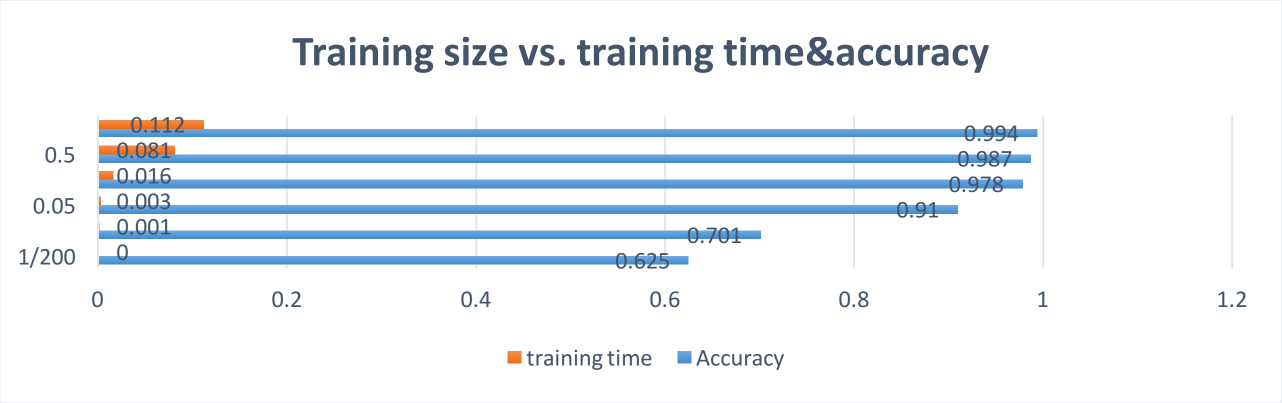


Bank Marketing – Neural networks

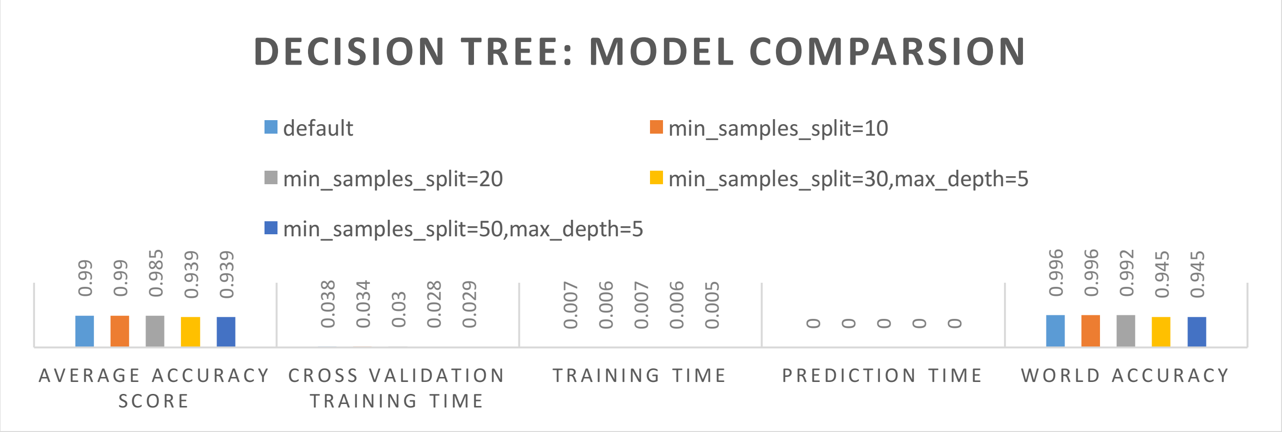
The MLPClassifier in sklearn is intended to be implemented in this step. Unfortunately, sklearn 0.17 doesn’t support multilayer perceptron. I intended to upgrade sklearn to 0.18 dev version, but Canopy has compatibility issue with the upgrade using git. Due to the time limit and technical capability restraint, this dataset is unable to transfer into arff format to be analyzed in GUI. Future research will be conducted in order to apply neural network algorithm on bank marketing data.

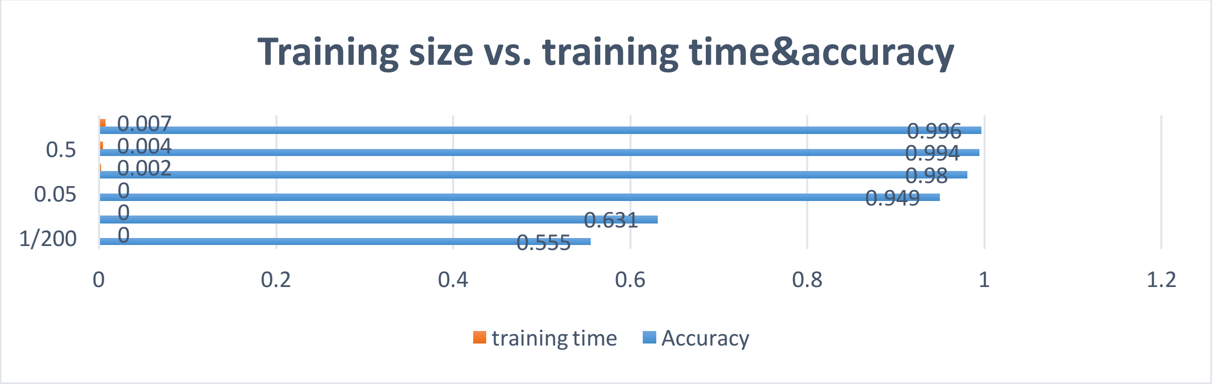
Chess – SVM



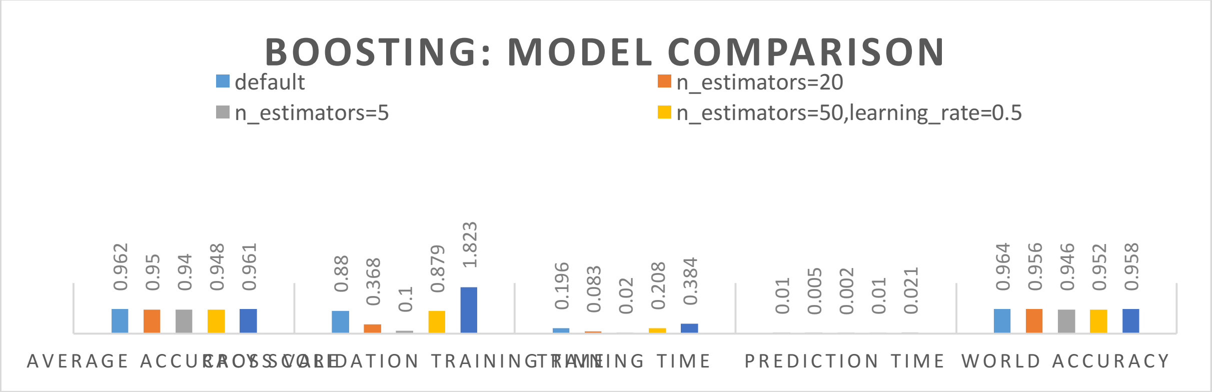


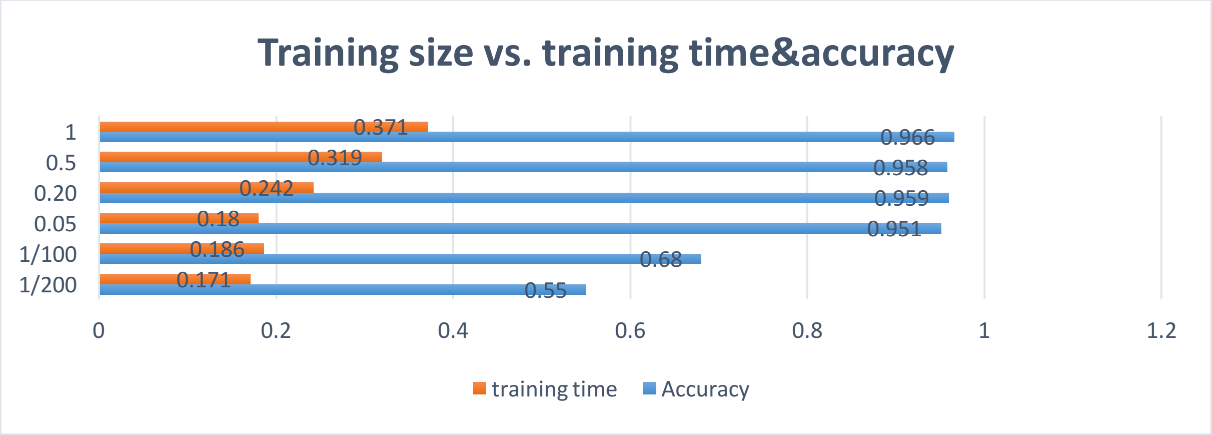
Chess - Decision Tree



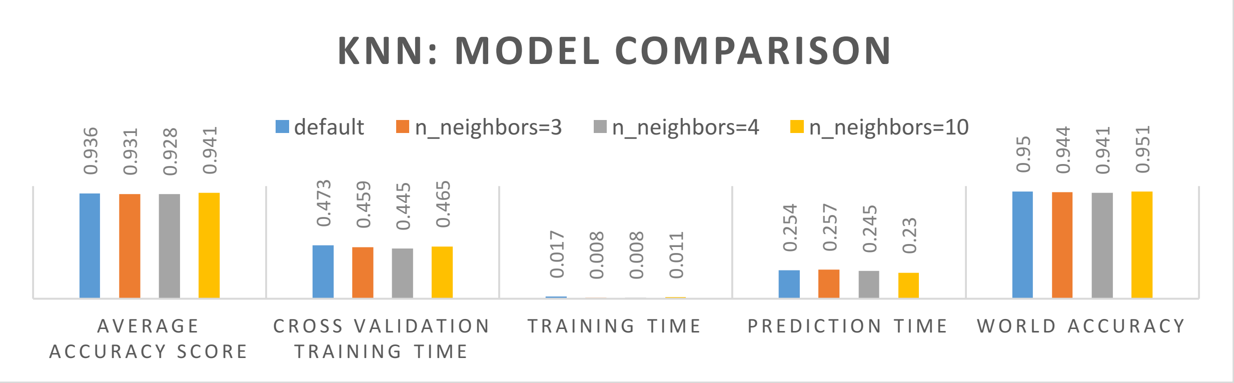


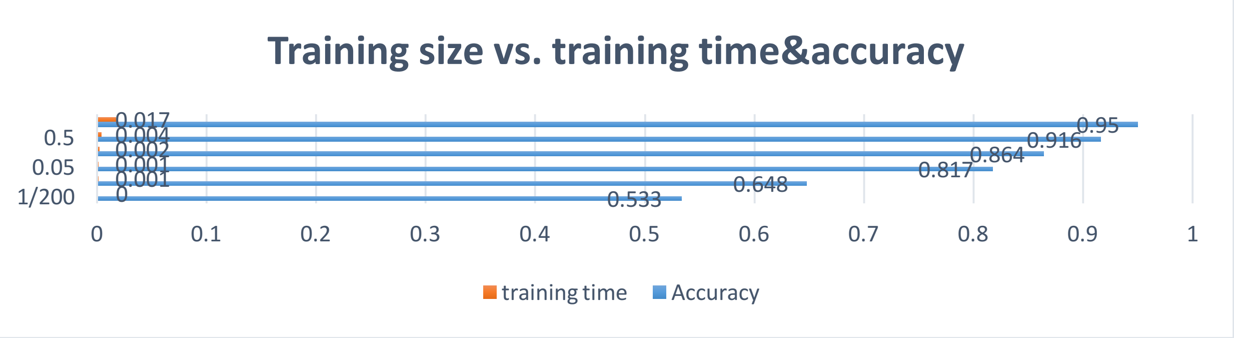
Chess – Boosting





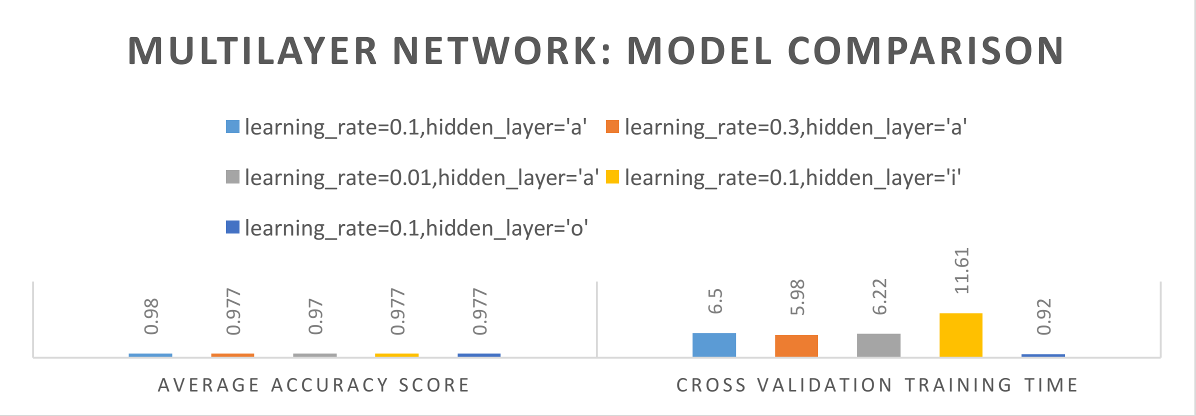
Chess – KNN

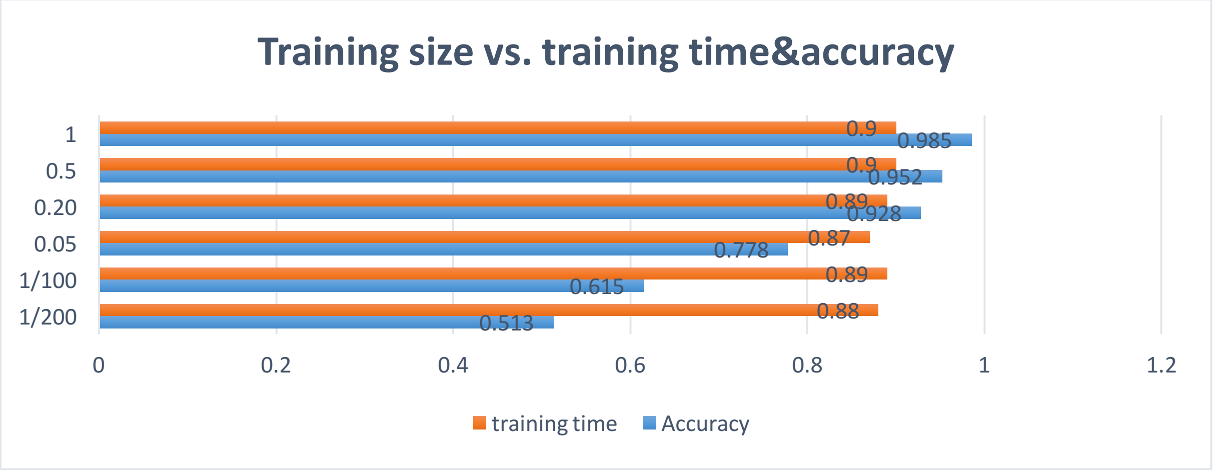




Chess – Neural Network

Due to the limit of sklearn version 0.17, MPLClassifier can’t be appied on the data. Therefore, weka GUI is used for testing neural network algorithm. Below are detailed data.





Analysis of result:

Speed and Accuracy analysis:

From the results above, it’s found that Support Vector Machine is the most computation heavy algorithm, especially for Bank marketing data. As mentioned before, in order to target similar prediction accuracy, SVM spends much more time than other algorithms. When kernel was tuned to be “linear”, the phenomena was fully exposed. In chess dataset, when “linear” kernel was implemented, it was 400 times slower than using “rbf” kernel. On the other hand, for chess dataset, the calculation time of SVM is much faster than SVM for bank marketing dataset. The reason may be that chess each chess attribute has less possible values. Therefore, SVM only needs to bring up minimum amount of dimensions.

Boosting, decision tree and KNN are much faster. Meantime, they generate about same accurate models as SVM, sometimes even better.

Comparing the performance of decision trees on two datasets, we can find that it indicates overfitting for bank marketing data. When min_samples_split becomes larger (minimum amount of nodes allowed for splitting) and max_depth becomes smaller (the maximum levels of tree), the accuracy tends to go up. That means by default the decision tree generates too many levels and split to too many branches to fit the training dataset. The resulted model doesn’t general pattern. On the other hand, the decision tree does very well with chess data set. By default, it renders a 99.6% prediction accuracy. By reducing the tree branch and leaves, its performance gets poorer. One reason is that chess board is very deterministic. When chess board is described by multiple attributes in enough detail, whether White can win or not becomes certain. Decision Tree is extremely good at describing deterministic events.

Boosting and KNN doesn’t have too much difference in performance. For both datasets, the more estimators (for boosting) or more neighbors (for KNN), the more accurate prediction will be. Of course, that means more training time. For boosting, it seems number of estimators and training time have a linear relationship. This relationship for K-nearest neighbors is not obvious.

Learning Curve analysis:

This part is to find the relationship between size of training data and model prediction accuracy. For chess data, 1/20 of the original data (3196 incidents) will bring the forecast accuracy to a reasonably good level. After that, increasing the number of data will slightly improve performance of the model. This holds true for all the tested algorithms.

However, for bank marketing data, an interesting phenomena was observed. No matter how small the size was, the model performance wouldn't be affected by too much for all the algorithms. One possible reason is that the chessboard can't be fully described by too few number of incidents. On the other hand, for bank dataset, several individual data can be used to represent majority of the population. Since whether people will decide to subscribe the service is a stochastic process. Therefore, models can generate decent prediction result based on a few data points but it will be very hard to surpass certain threshold, 95% for example.

Neural network is also a very accurate algorithm with a little longer training time. For five-fold cross validation, it takes on average 5 seconds to form the data. However, if hidden layer 'o' is applied, the training time is magnificently decreased with the accuracy remain unchanged. This can be a good selection when speed is a major concern in the project.

Overall, decision tree is the best algorithm for both datasets. Not only does decision tree require minimum amount of training time and prediction time, but it also maintains as one of the algorithms that has the highest prediction accuracy. When given minimum viable amount of training data, decision tree gets close to optimal forecasting accuracy very fast.

Improvements on models:

Many things could be done to improve the accuracy of the algorithms. In general, if more time is allowed and more computation power is accessible, I would apply GridSearchCV, which is a nice little widget we can use to automatically adjust the parameters in the algorithm to find out the best tuned model for certain algorithm. This will give me more stringent selections of parameter settings instead of trying different settings manually. For instance, for decision tree, instead of guessing the most appropriate min_samples_split or max_depth, if I throw in a list of parameters, the system will generate the best-fitting pruning metrics in a systematic manner.

More robust cross validation should be applied when time permits. Currently for all the algorithms except SVM I applied 5-fold cross validation. A higher order-fold cross validation will minimize the variance and avoid the situation where data is not shuffled and split random enough. This can become an issue, especially when dataset is not big enough.

For Support Vector Machine, due to the limit of the computing power, I was only able to do three-fold for the cross validation. Although it eliminates some of the bias, a more divided cross validation dataset is preferred. One way to solve the computing issue is to run codes in virtual machines over cloud service, like AWS or Azure. More research will be conducted on how to do that. Another issue is that system always crashes when I try to run ‘linear’ kernel on the dataset. For boosting, if different forms of weak learners can be specified, the prediction accuracy can be improved and less learners will be needed.

Moreover, as only neural network is tested using weka GUI, the configuration between sklearn and weka might be different and hence result in different result. Given more time, I would test all the other four algorithms on weka and see whether there will be performance change.

In order to understand all the five algorithms more thoroughly, classification problems with more than two categories can be experimented in the future. Both datasets only ask for the classifier to categorize the data into two categories. For bank marketing data, it’s whether customers will subscribe; for chess data, it’s whether White will win or not. In reality, a machine learning classification problem can be much more complicated with more categories. For instance, depending on the environment, an intelligent robot should be able to respond with actions that’s more than just picking two options. For example, for a simple two category problem decision tree can do really well, but maybe neural network will beat all other algorithms when solving a more complex problem.