CS4641 Machine Learning

Assignment 1: Supervised Learning

Kisung Park

1. Overview

The purpose of this assignment was to try applying multiple supervised learning methods to dataset and analyzing the result. In the meantime, I tried to tune mutiple hyperparameters for each methods for best accuracy and find the best model for the dataset. I used two kinds of datasets both of which were obtained from kaggle.com

1) Graduate Admission Dataset

This dataset represents a small dataset with relatively small number of features and small size. It has 7 features and only 500 rows. It shows the chance of the admission for each applicants with their GRE score, TOEFL score, University rating, SOP, LOR, CGPA, and Research experience. Also, the label, the chance of admission is given as a probability, requiring division into certain range for classification.

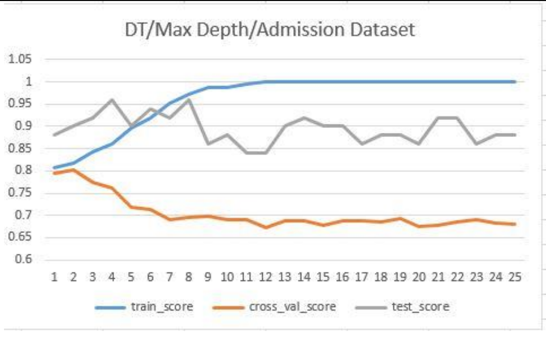
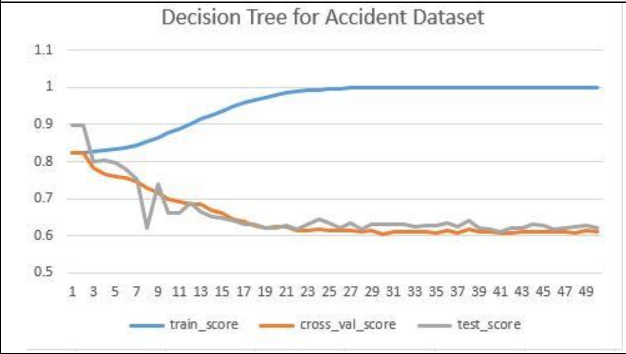
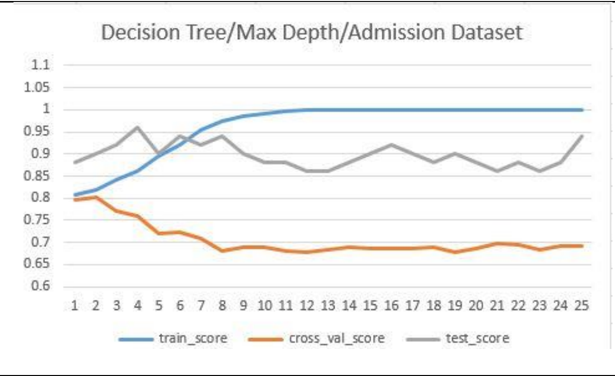
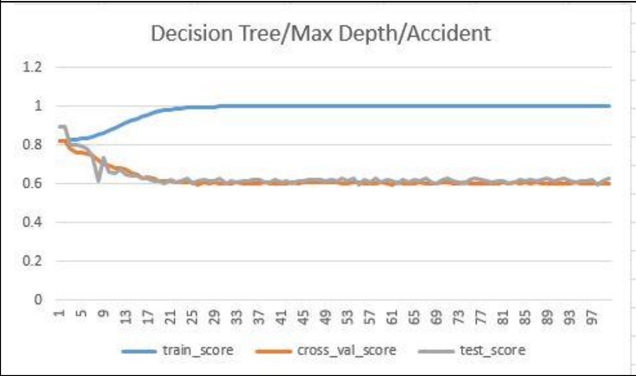
2) UK Car Accident Dataset

The other dataset represents big dataset with relatively big number of features and big size compared to the previous one. It has 14 features and 10000 rows. The actual dataset is consist of 30 features and 1000000 rows, but some of the features such as police attendence or local authority showed small variant, so I selected 14 features that seemed to be important. Geographic status(Longitude, Latitude, road number, urban/rural area), time conditions(day of week, time of day), driving condition(road surface condition, light condition, weather condition, speed limit) were used as features and the label was the severity of accident which was labeled between 1 to 3. The longitude and the latittude was recorded in float and all the other features and label was marked in integer.

Analysis of each dataset will be performed using the five algorithms presented. or the analysis of model complexity, the training and test datasets will be partitioned from the full data in a **9:1 ratio**, respectively. For cross validation on training datasets, **10 folds** of partitioning will be applied. Furthermore, to find out the effect of pre-processing, the model complexity analysis was conducted twice, each **with or without normalization**. The analysis will be performed firstly on **model complexity curves**, and with the appropriately tuned hyperparameters, analysis of **learning curves** would follow.

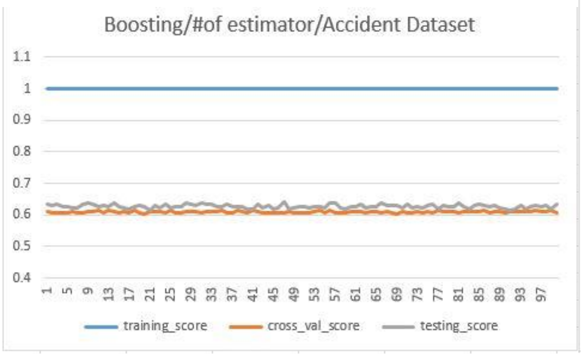
2. Model Complexity Anaysis

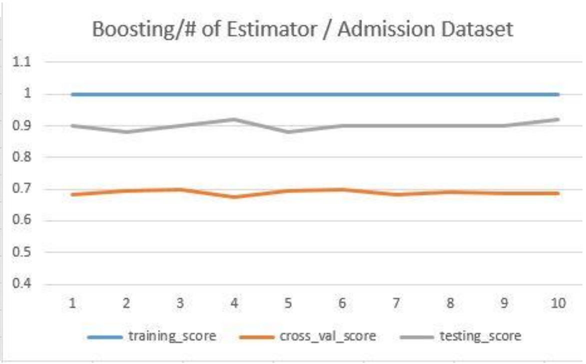
1) Decision Tree

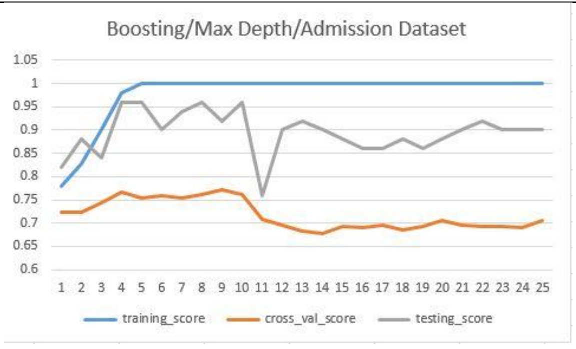
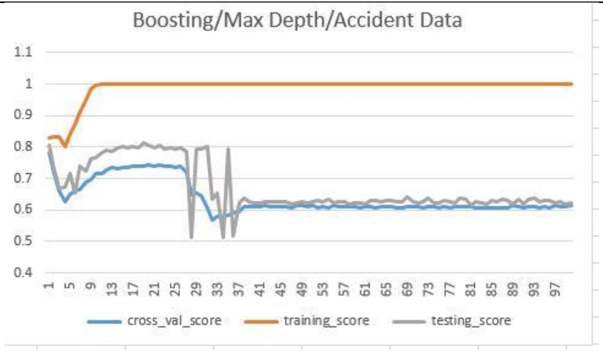
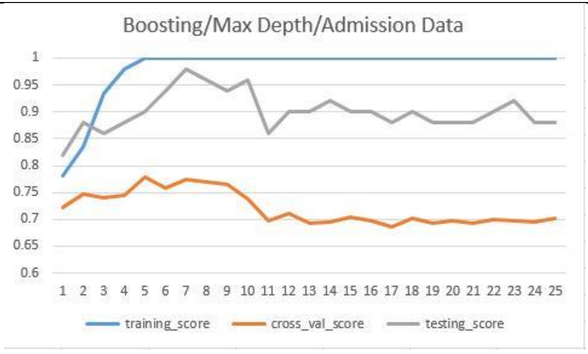
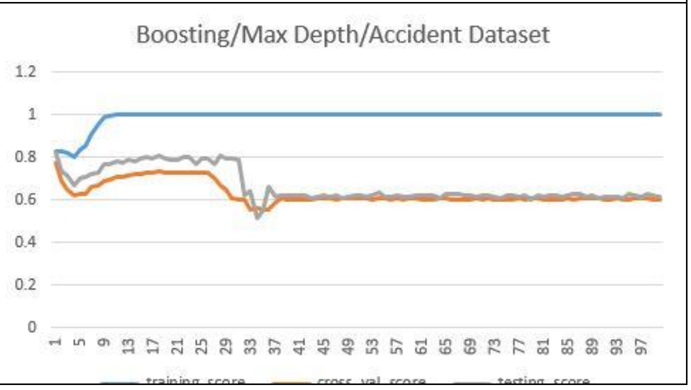
	Admission Dataset	Accident Dataset
Without Normalization		
With Normalization		

I used sklearn library's DecisionTreeClassifier function to implement decision tree supervised learning. The pruning method applied for my implementation was maximum depth or the height of the tree. Limiting the height of the tree can help the model avoid overfitting. **The optimal height of the tree can be determined as 4 and 2** for each dataset, as the training score starts to rise, however the cross validation and the test score drops. Which implies that the model gets overfitted. Also, it shows that the normalization didn't affect the overall performance. This could be expected from the fact that decision tree will intuitively develop attributes related to input features. Applying normalization will result in the similar tree only with normalized attributes.

2) Boosting

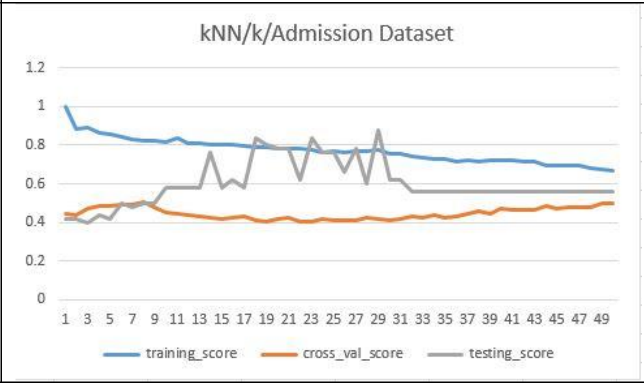
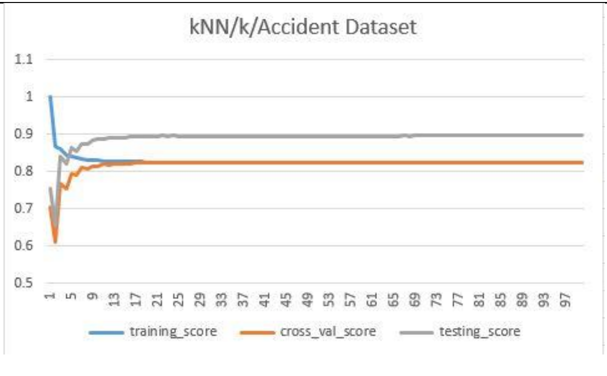


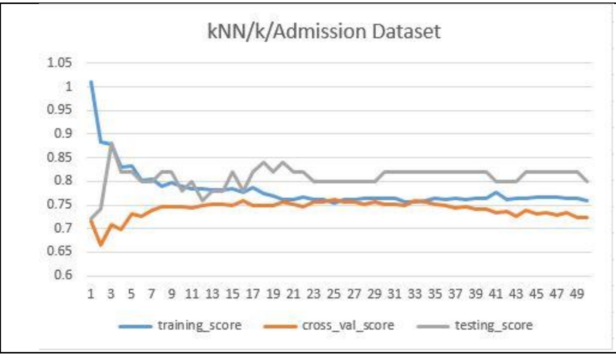
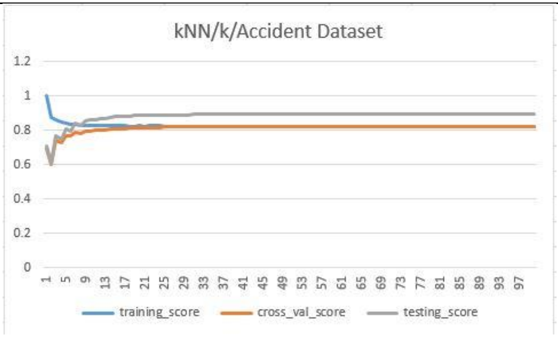


	Admission Dataset	Accident Dataset
Without Normalization		
With Normalization		

I used sklearn library's DecisionTreeClassifier and AdaBoostingClassifier function to implement decision tree supervised learning. AdaBoosting make use of ensemble of multiple estimators, so I modified the number of estimators to see the change. However, the number of estimators did not affect the accuracy for decision tree with specific height. Just like the normal decision tree, the pruning method applied for my implementation was maximum depth or the height of the tree. **The optimal height of the tree can be determined as 6 and 14** for each dataset. This number is larger than the normal decision tree as we could expect more aggressive pruning. Also, it shows that the normalization reduced the overfitting in decision tree with higher height.

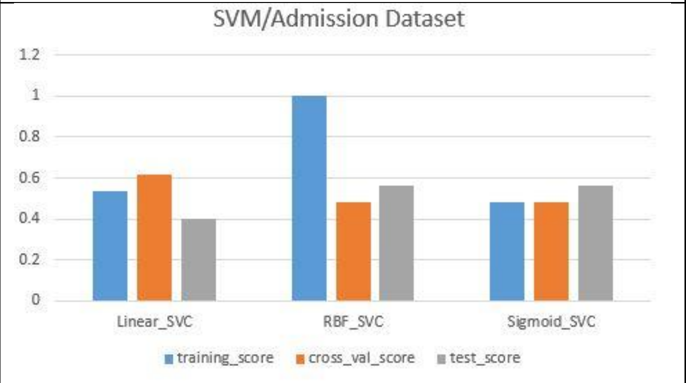
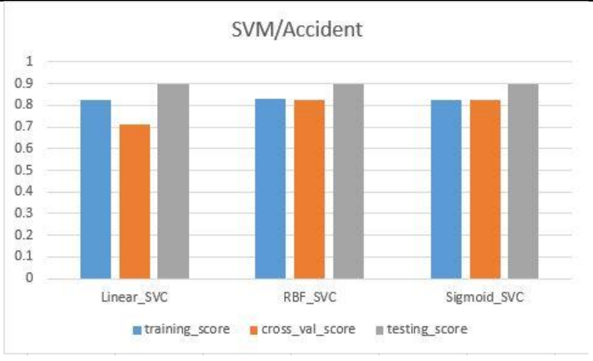
3) kNN

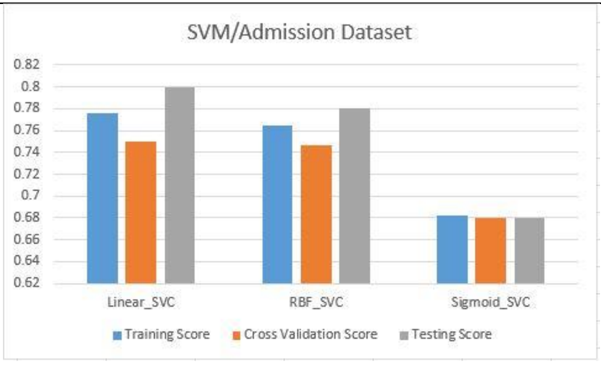
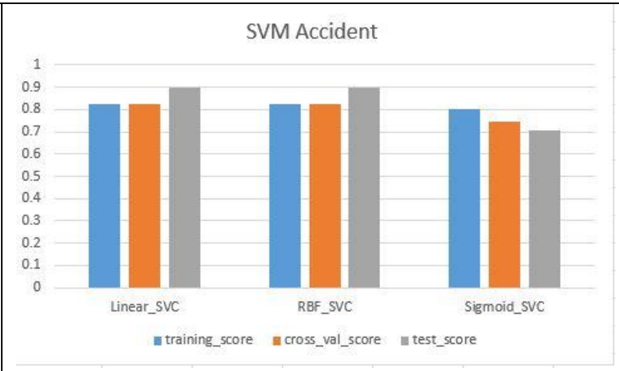
	Admission Dataset	Accident Dataset
Without Normalization		

With Normalization		

I implemented my own code for kNN supervised learning. I confirmed my implementation by comparing with the sklearn's kNN implementation and they gave the equal result. I measured the accuracy with number of k changing. For Admission dataset, the graph saturated at around **k = 30** and for accident dataset, the graph saturated at around **k = 20**. For Accident dataset, normalization had no effect. The feature with large absolute value for accident dataset was longitude and latitude, but the scale of difference among the values was less than 5, which is in the same range as other features. However, the **normalization affected highly** for admission dataset for this method. Without normalization, there was more than 0.2 of accuracy drop for cross validation and testing score. This is because the pure kNN uses distance measure for finding the nearest neighbor, and each of the features are not weighted. Therefore, if the absolute value for a feature is bigger than the others, it will affect the distance more than other features although the variance of the feature is low. Therefore, normalizing by dividing the maximum value should be applied for this method of supervised learning.

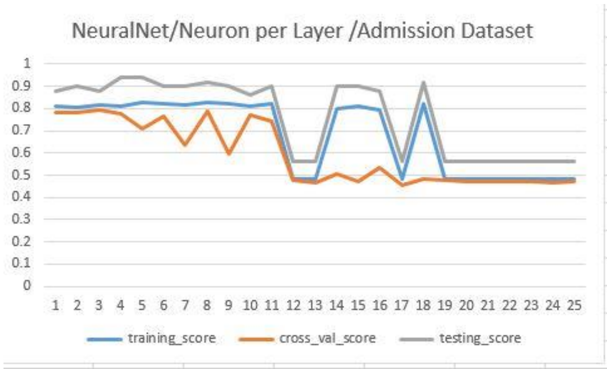
4) SVM

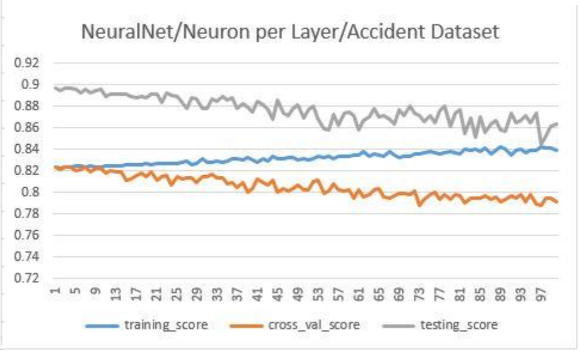
	Admission Dataset	Accident Dataset
Without Normalization		

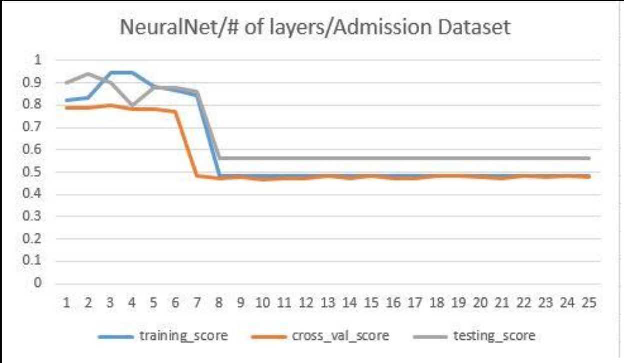
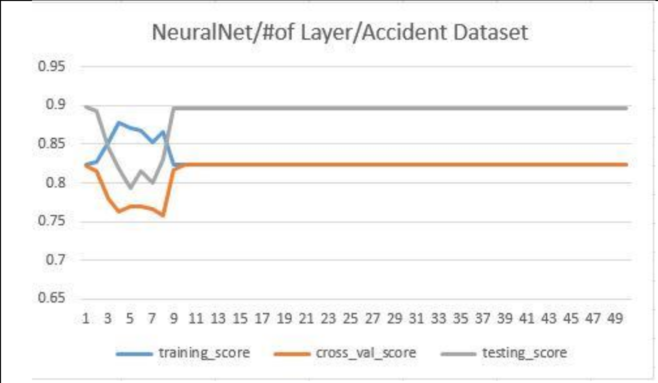
With Normalization		

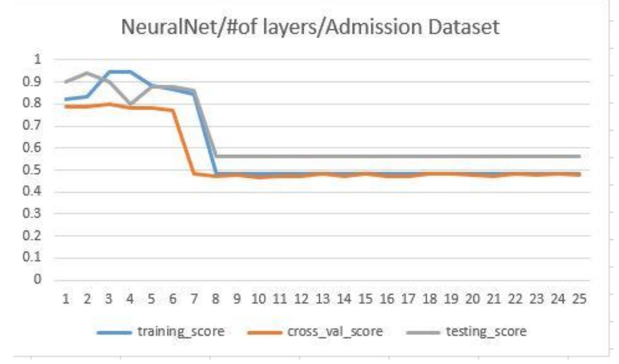
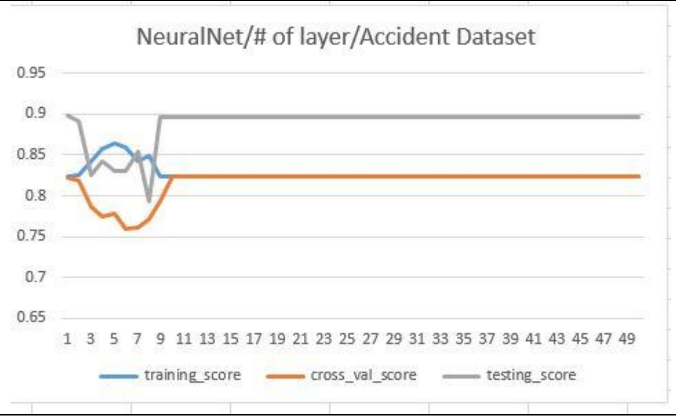
I used sklearn library's SVC function to implement decision tree supervised learning. For admission dataset, it turned out that using **linear kernel function** showed best accuracy. Also, normalizing increased the accuracy for all three kernel functions. However, for accident dataset, **all three kernel functions** worked well.

5) NeuralNet





	Admission Dataset	Accident Dataset
Without Normalization		

With Normalization		

I used sklearn library's MLPClassifier function(which represents multi-layer perceptron classifier) to implement decision tree supervised learning. I used Adam optimizer and se the learning rate to 10^-5. First, I measured accuracy according to the number of neurons for each layer. For admission dataset, using less than **10** neurons per layer showed good performance. For accident dataset, large number of neurons led to overfitting. So I determined on **9** for the number of neurons.

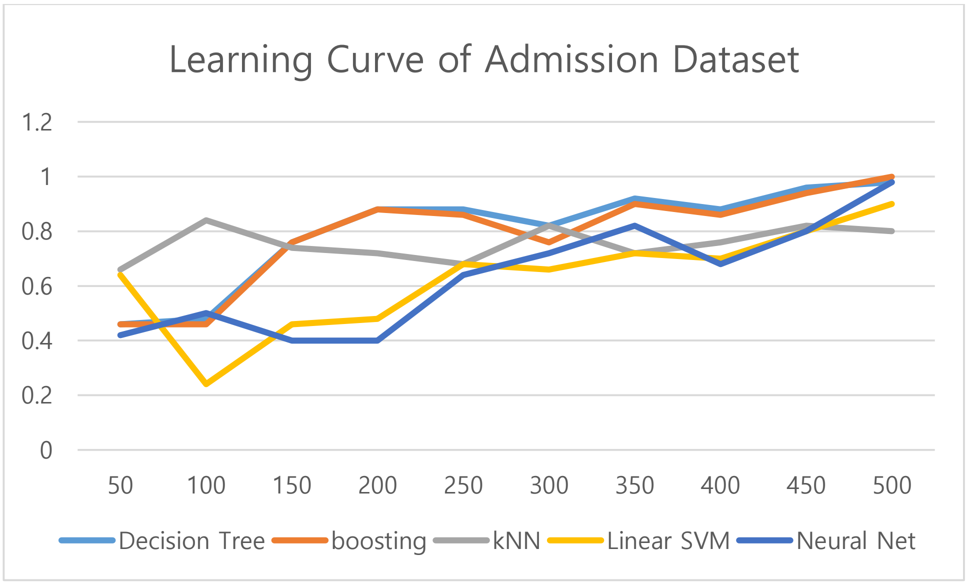
With optimal number of neurons per layer, I measured the accuracy according to the number of layers for neural net. For admission dataset, it turned out that the accuracy drops if the layer gets deeper than more than **6**. However, for accident dataset, the accuracy increased and saturated as the layer gets deeper, with more than **11** layers. Normalization did not affect the performance which could be expected from the fact that neural net weights each perceptron that the weight is trained by the datasets.

3. Learning Curve

	Admission	Accident
DT	Max depth = 4	Max depth = 2
Boost	Max depth = 6	Max depth = 14
kNN	K = 30	K = 20
SVM	Kernel function = Linear	Kernel function = Linear, RBF, Sigmoid
NN	#of neurons =10 # of layers = 6	# of neurons = 9 # of layers = 11

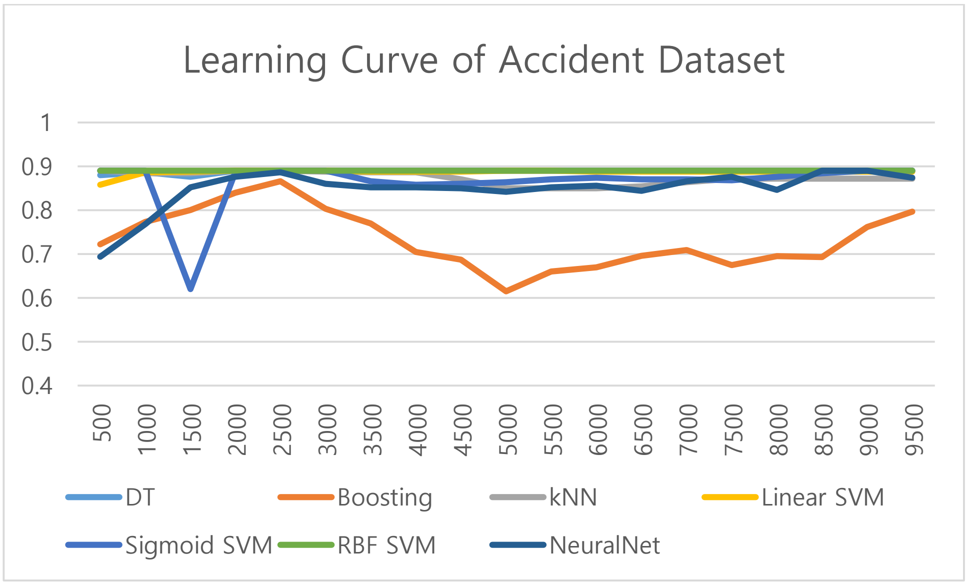
Above is the table of optimal hyperparameters that I found out from the model complexity analysis. I used these parameters to get the learning curve. For the admission dataset, I used 500 training dataset and 50 validation dataset. I increased the number of training dataset from 50 to 500 by 50 at each iteration and obtained the learning curve. For the accident dataset, I used 9500 training dataset and 500 validation dataset. I increased the number of training dataset from 500 to 9500 by 500 at each iteration and obtained the learning curve.

1) Admission Dataset



The admission dataset is a small sized dataset, so it was expected that there the speed of accuracy growing is fast. This phenomenon appeared in most of the methods. However, for kNN, the accuracy was reasonably high even with small number of dataset. All of the models achieved accuracy over 0.8 after fully trained. The learning curve did not always showed growth. There was a drop of accuracy in some stage. This is because of the size of the dataset is small, so even a small bias can train the model toward far from the correct direction. Still, the dataset it well formed, so it showed good performance with enough number of training dataset.

2) Accident Dataset



The accident dataset has more features, so it was expected to show good accuracy from the start. Models such as Linear SVM or RBF SVM showed good performance even with a small number of dataset. Neural network was the model that showed some growth as the number of training dataset gets larger. All of the models achieved accuracy over 0.8 after fully trained.

All of the models except for the boosting converged to accuracy of 0.9 after using 2000 training dataset. However, boosting showed some drop in accuracy when trained with 2500 dataset and increased again when trained with 5000 dataset. The reason for this behavior is assumed to be the uneven distribution of dataset. Our dataset is labeled as 1 to 3 for the severity of the accident. Among these three, the data labeled as 1 was less popular and when I looked through the real dataset, most were concentrated at the front and the back portion of the dataset.

4. Conclusion

From the analysis, we found out that some models are more appropriate for small dataset than the others. Decision tree and Neural network seemed to be inappropriate if we have a limited number of dataset. Whereas kNN worked well even with a small number of dataset. However, kNN failed to show the best performance if we have enough training dataset. Also, we found out that normalization can affect the performance of some specific models. We found out that kNN is not appropriate for small dataset without normalization. Not only is the accuracy but also the time consumed for training should be considered. The training time of each models is compared as Decision Tree = kNN < Boosting < SVM << Neural Network. Considering these results, we can characterize each models as below.

Decision Tree: Fast trained, but requires enough training dataset. Pruning required to avoid overfitting.

Boosting: Requires enough training dataset. Pruning required to avoid overfitting.

kNN: Fast trained, works well with small number of dataset, but not a choice if we want the best performance.

SVM: A general method that is applicable regardless of the dataset size but still takes time.

Neural Network: Takes a long time to train, not good with limited number of dataset, more appropriate for complicated dataset with many features.