Assignment 1 : Supervised Learning

Sameer Ahmad SIddiqui
ssiddiqui39
CS-7641 - Spring 2021

Datasets Description:

Dataset 1: Bankruptcy data

It is a dataset of Bankruptcy data from the Taiwan Economic Journal for the years 1999–2009 for prediction of companies going to file for bankruptcy or not. Data has 6819 rows and 25% split for testing. It is picked from kaggle. Dataset has 95 columns like total income/Total expense, Current Liability to Equity and No-credit Interval and many more. It also has 1 label for classification for bankruptcy? Which have binary classes. I am interested to find out how accurately we can predict bankruptcy of a company and which features do contribute more towards such prediction. Which algorithm would provide best results.

This dataset had all the features in float and int format and label was binary, hence it did not need much of the preprocessing and was ready to be read and processed

Dataset 2: Happiness index data

It is a dataset which captures the basic economic and social features of a country. Dataset is picked from Kaggle and the dataset from multiple years is combined to learn the happiness index over the period of 2018 and 2019. Data has multiple labels for happiness score ranging from 1 to 10 and it has 5 features namely GDP per capita, Social support, Healthy life expectancy, Freedom to make life choices, Generosity. It would be interesting to check the importance of economic factors on the happiness of the population. This dataset is fairly small with 312 rows across 2 years of data.

This dataset needed more preprocessing, which needed Encoding for region and since the label was in float it needed to be converted to int for classifications algorithms.

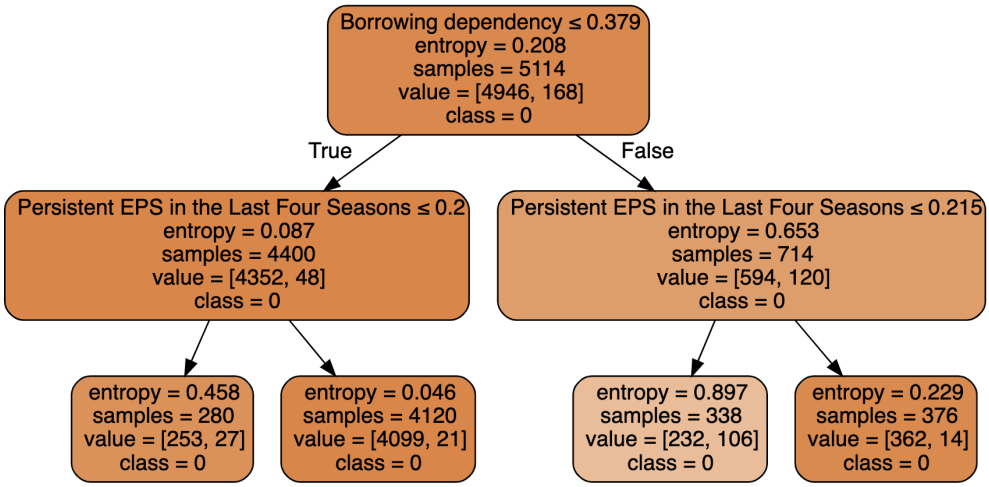
Decision Trees

I started by using the grid search algorithm to find the most appropriate hyper parameters to use in the algorithm. For decision tree classifiers, I am using the six hyper parameters in the grid search. These are criterion, max_depth, max_features, max_leaf_nodes, min_samples_split, random_state. Grid search resulted in following as best param value.

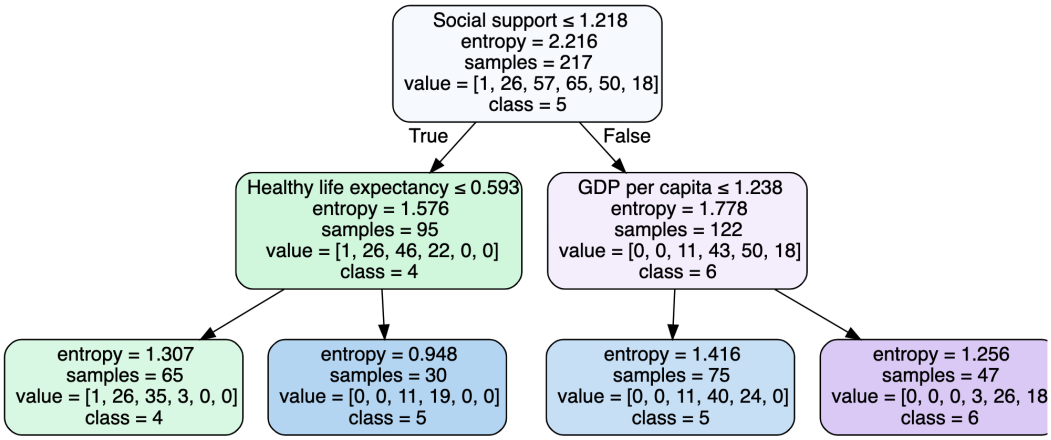
Best params	Bankruptcy data set	Happiness index dataset
'criterion'	entropy	entropy
'max_depth'	4	3
'max_features'	log2	None
'max_leaf_nodes'	10	10

'min_samples_split'	2	2
'random_state'	3	1
Best score	0.9698855865006992	0.6450317124735729
Time (seconds)	259	36

Visualizing the graph with max depth of 2 for Bankruptcy dataset

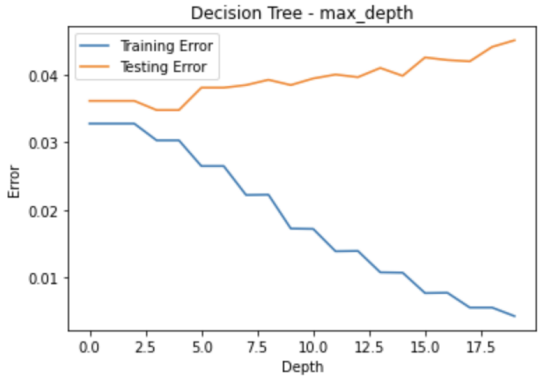


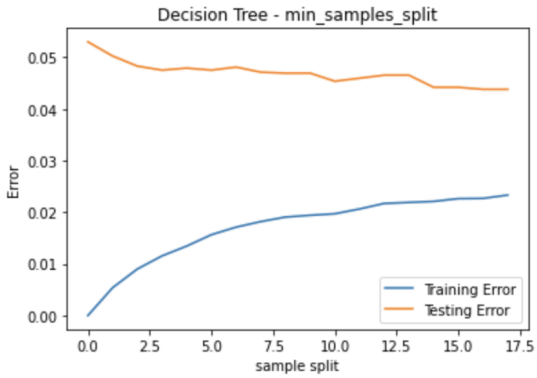
Visualizing the graph with max depth of 2 for Happiness index dataset



I used max_depth and min_samples_split to validate the results from grid search. Max_depth validation curve shows for training data , as the depth of the tree increases, the error tends to approach 0 however the same is not true for the testing error. Error for testing increases with increasing depth of the tree which is demonstrating overfitting situation.

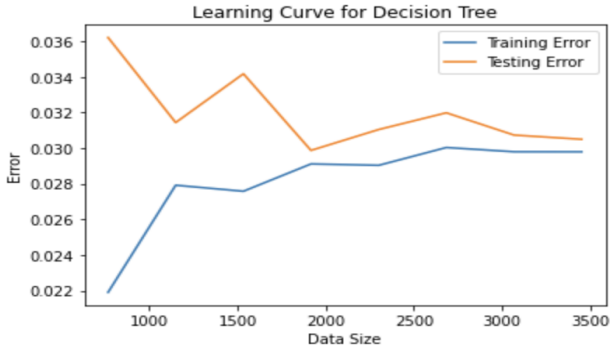
Bankruptcy dataset - Validation curve:





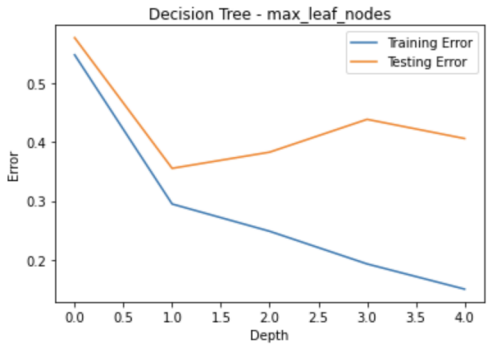
Validation curve for sample splits shows convergence with increase in number of sample splits. By increasing the sample split , we would be able to generalize the data and make accurate predictions. Learning curve for bankruptcy data has almost converged after variations initially, variances have reduced and training score is largely generalised.

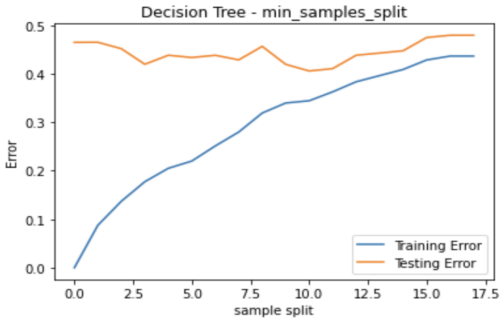
Bankruptcy dataset - learning curve:



For the happiness index datasets, I am using max_leaf_nodes and min_sample_splits to validate the results from grid search. With increase in depth, error for training data is approaching low and I can assume that increase in leaves count improves the error rate for training data but not necessarily the testing data.

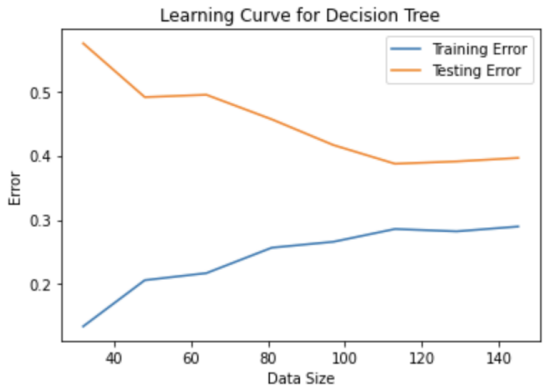
Happiness index dataset - Validation curve:





Happiness index dataset - learning curve:

Decision tree is reacting to variations in data and generalizing the training score. But overall accuracy is not increasing more than 0.6.



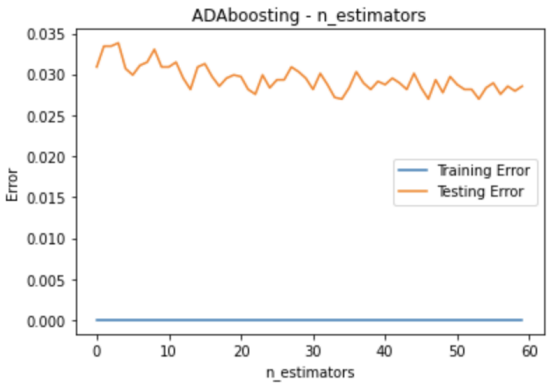
Boosting

For ADAboosting classifier, to perform grid search , 3 hyper parameters are used, these are learning_rate, random_rate, n_estimators. It uses a decision tree classifier with max_depth and min_samples_split values from the best params of Decision tree grid search, which gives better overall performance. Below are the values from the grid search result.

Best params	Bankruptcy data set	Happiness index dataset
'learning_rate'	1.0	0.75
'n_estimators'	50	50
'random_state'	3	3
Best score	0.9739640344018765	0.6018181818181818
Time (seconds)	1203.118	20.74

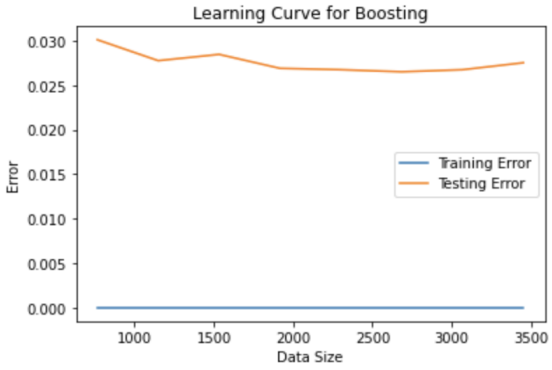
Bankruptcy dataset - Validation curve:

As the boosting algorithm works on multiple subsets of data and modifies the weights also, n_estimators (Number of estimators) have a great level of impact on error rate. I have noticed the training error remains fairly constant but error rate increases for testing data increases with increase in learning rate.

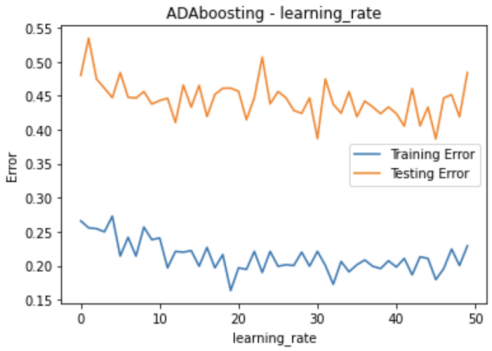


Bankruptcy dataset - learning curve:

As the dataset or number of rows increases, the error rate for testing data increases again, which is unique behaviour.

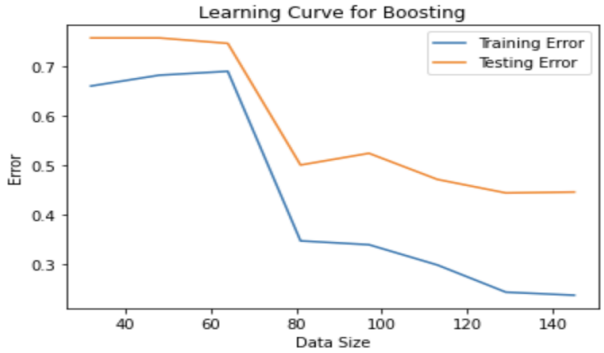


Happiness index dataset - Validation curve:



On the smaller datasets, the learning curve shows that, with increase in data, convergence occurs.

Happiness index dataset - learning curve:



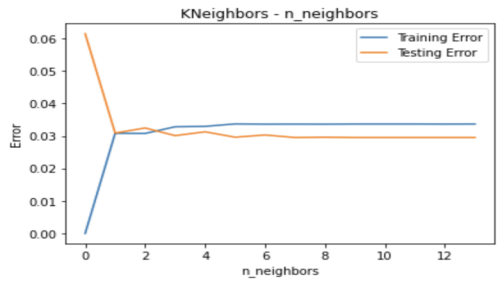
k-Nearest Neighbors

I started the grid search with 3 hyper parameters which are *algorithm, n_neigbors, leaf_size*.
For algorithms, I tried *auto, ball_tree, kd_tree* and *brute.* For other parameters, tried the values based on data sizes also

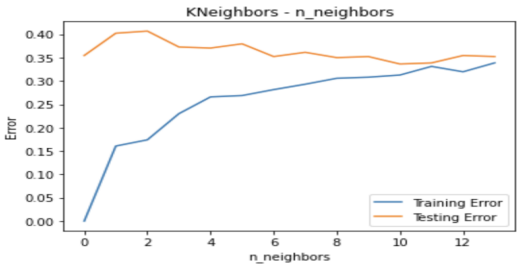
Best params	Bankruptcy data set	Happiness index dataset
'algorithm'	auto	auto
'leaf_size'	4	4
'n_neighbors'	10	1
Best score	0.9671491124871594	0.6638477801268499
Time (seconds)	55	3

Performance of the model is compared based on the number of neighbors, what I notice is that for bankruptcy dataset which is larger dataset, shows error rate is fairly constant after reaching certain number of neighbors, while for the smaller dataset when number of neighbors increases after certain point , error rate becomes slightly constant and converge.

Bankruptcy dataset - Validation curve:

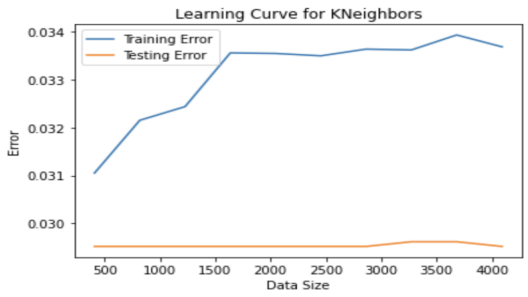


Happiness Index dataset - Validation curve:

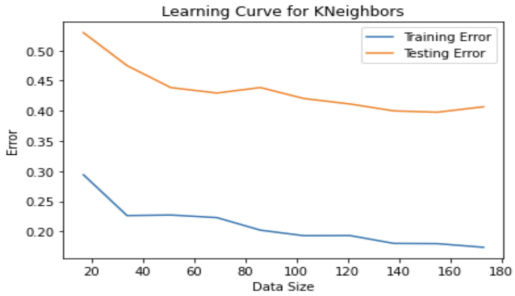


The learning curve for the KNN shows both the datasets , larger bankruptcy data and smaller happiness index data. One Important point to notice is that curves do not converge. It can be inferred that additional data for training may not improve the performance any further.

Bankruptcy dataset - learning curve:



Happiness index dataset - learning curve:



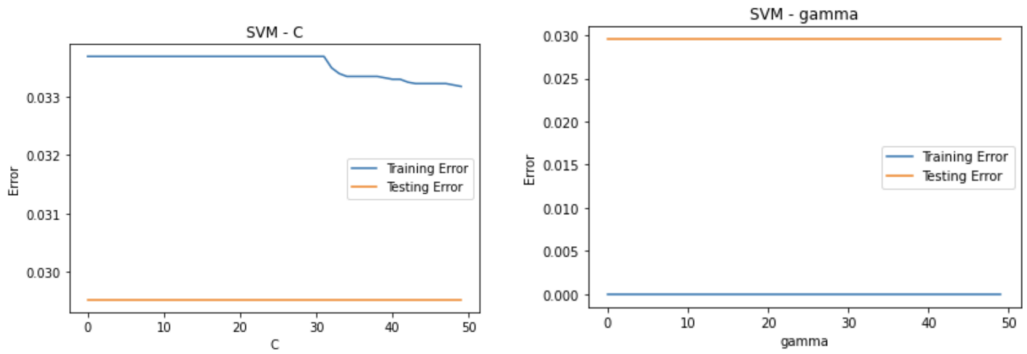
Support Vector Machines

To perform the grid search for the SVM, I started with 3 hyper parameters, which are *gamma, C* and *kernel.* For the kernel, I have tried both linear and gaussian functions. By running brute force grid search, following are the values which are the best parameters for the estimator.

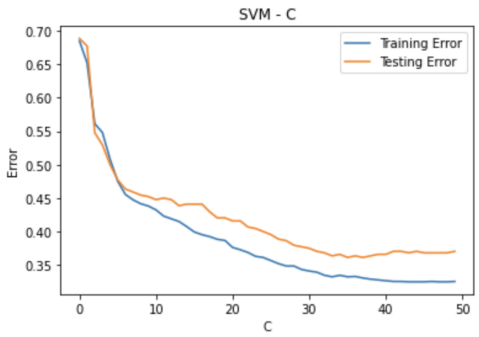
Best params	Bankruptcy data set	Happiness index dataset
'C'	1.0	1.0
'gamma'	0.1	0.163
'kernel'	'rbf'	'rbf'
Best score	0.9675403106247119	0.6408033826638478
Time(sec)	312.649	1.90

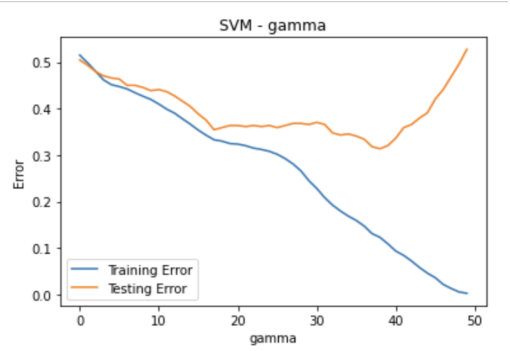
Validation curve shows that C has an impact on the performance of the algorithm and impact is profound on smaller datasets. Error rate comes down for both training and testing data specially for smaller dataset. Gamma had limited impact on larger dataset but on smaller dataset , we can notice error rate of the testing data increases with increase in gamma.

Bankruptcy dataset - Validation curve:



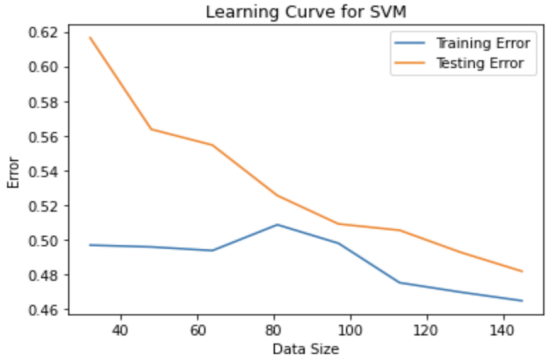
Happiness Index dataset - Validation curve:





Happiness Index dataset - learning curve:

The learning curve shows convergence and error rate decreases with increasing data.



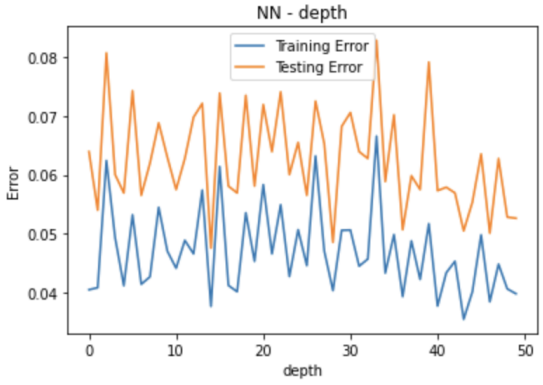
Neural Networks

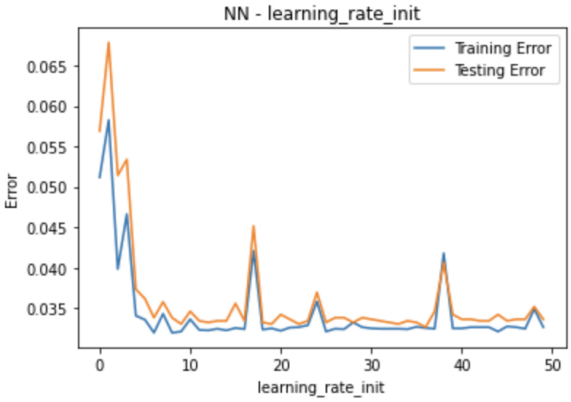
I started the grid search for a neural network algorithm with 5 hyper parameters, these are *activation, alpha, learning_rate, learning_rate_init, hidden_layer_sizes* , range of the values is chosen obviously based on size of data as well. Below are results for the best estimators.

Best params	Bankruptcy data set	Happiness index dataset
activation	tanh	tanh
hidden_layer_sizes	10	4
alpha	0.001	0.0001
learning_rate_init	0.001	0.01
learning_rate	invscaling	constant
Best score	0.967344807203402	0.6455602536997886
Time(sec)	795.34	333.55

Bankruptcy dataset - Validation curve:

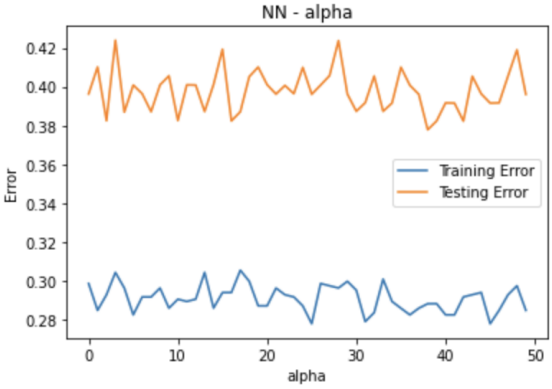
Two factors which impact the error rate for NN performance, are *alpha* and *learning_rate_init*





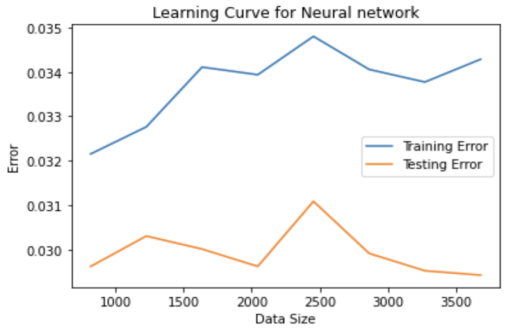
Example of the impact of alpha shows that the model fits pretty well to both training data and testing data. Whereas, looking into learning rate init, error rate drastically reduces with increase in depth of the data.

Happiness Index dataset - Validation curve:

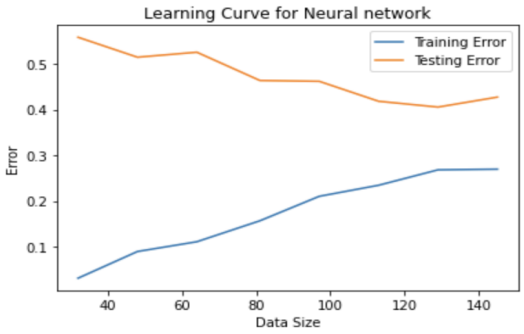


Looking at the learning curve shows as data increases . The model is fit much better when data is small as error rate tends to move towards 0 for the larger dataset of bankruptcy data, while it is not true about the smaller dataset.

Bankruptcy dataset - learning curve:



Happiness Index dataset - learning curve:



Analysis:

Below is a list of the performance in terms of accuracy and time spent. Performance of the algorithm is in similar range for both datasets. Range of performance for the large dataset(~2k) is around ~96 % while the same for the smaller datasets(~ 200 rows) performs much lower around the same range of 65%.

To begin with, the larger dataset of bankruptcy which is from the Taiwan Economic Journal for the years 1999–2009. On a large scale all of the algorithms perform similarly. In terms of score, ADAboosting performs the best with 97.39 % which is a good score for larger dataset. In terms of time KNN had best performance for the Bankruptcy dataset. The performance of NN was fairly similar to all other algorithms but looking at the learning curve, it appears more data is not going to improve the results for this dataset. Learning time for KNN was very small compared to other algorithms which can be attributed to the lazy learner method of KNN. Decision tree performance was slightly improved by boosting while time performance has really improved by more than 50%. Though more data would not have helped improving the accuracy as the error rate is not converging.
For the smaller dataset, which is the dataset which captures the basic economic and social features of a country and happiness index is calculated based on economic and social factors. The best performance is from KNN which gave close to 66.3 % results. In terms of time, the best performance was from SVM with slightly lesser than 2s. Neural networks have hugely large training time compared to other algorithms, it can be attributed to the eager learner nature of the algorithm. Learning curve of NN suggests that accuracy of the algorithm would improve with more data to train on. SVM also has an error rate falling with more data and performance improves with more data.

Overall I believe, Decision trees with boosting performed the best for bankruptcy dataset and KNN performed best for happiness index dataset.

	Bankruptcy Data		**Happiness Index Data**	
	Score	Time(sec)	Score	Time(sec)
Decision Tree	0.9698	259	0.645	36
Boosting	0.9739	120.3	0.601	20.74
k-Nearest Neighbors	0.9671	55	0.663	3
Support Vector Machines	0.9675	312.64	0.640	1.90
Neural Networks	0.9673	795.34	0.645	333.55

Execution Environment:

References

1. Bankruptcy data from the Taiwan Economic Journal for the years 1999–2009 https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction
2. World Happiness Report https://www.kaggle.com/unsdsn/world-happiness
3. Scikit Homepage https://scikit-learn.org/stable/
4. Simple imputer https://datascience.stackexchange.com/questions/51890/how-to-use-simpleimputer-class-to-replace-missing-values-with-mean-values-using