KE5206 NEURAL NETWORK CA

Bui Kim Dung (E0146998)

SUMMARY

This CA project is to fit different neural network models and ensemble models on 2 datasets using 2 different tools:

- Dataset whitewine_quality: used Python sklearn library, trained MLP model with tanh and relu functions, then ensemble using BaggingClassifier and VotingClassifier.
- Dataset diabetes: used R with caret, caretEnsemble, nnet, elmNN packages, trained single layer neural network and extreme learning model, ensemble using Stochastic Gradient Boosting.

MODELING

I have applied following process for modeling both datasets:

1. Separate the dataset into training set and test set
2. Data preprocess using standard scale and center
3. Train and tune parameters for each NN
4. Run prediction and evaluate models using cross-validation
5. Create ensemble model and compare with individual models

Dataset: diabetes

Analysis on dataset

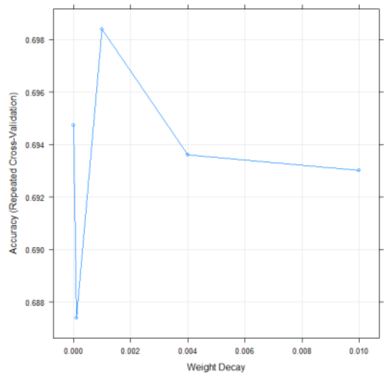
This is a relative small dataset with 798 samples of 9 variables. The classify variable has only 2 values 0 and 1 which indicates healthy or diabetes. All other variables values are in numeric and there is no NA value, so it is quite convenience to scale and center data. With the number of variables relatively small in compare with sample size, it is good to use a small size neural network. A quick training using MLP model with single hidden layer, 8 units, mini-batch training SGD using Python can achieve ~ 78% test accuracy.

Training tool and individual neural networks

R caret package (short for classification and regression training) contains functions to streamline the model training process for complex classification and regression problems. It contains all the necessary tools for data splitting, pre-processing, model tuning …

R caretEnsemble provides tools to create and train ensemble models: caretList to build lists of caret models and caretEnsemble and caretStack to create ensemble models from caretList.

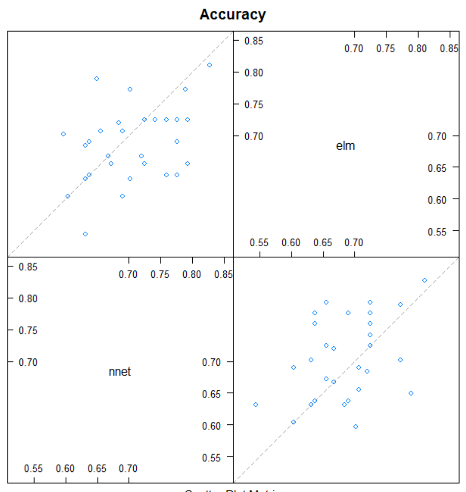
Single layer neural network: using nnet library with support of caret, 10-fold cross-validation, with tuning on size of the layer and decay variable, I was able to achieved ~76% accuracy.



Extreme learning model (ELM): is a simple neural network with randomize input weights and bias, using an inverse matrix to calculate output weight. This model could achieved 70% accuracy on training set, but 77% accuracy on testing set.

Ensemble model

A simple ensemble model Average Neural network did not give out much better result (~ 75% accuracy on training and ~ 77.6% accuracy on test set).



Scatter Plot Matrix

Calculate correlation score between the 2 individual NN resulted in 0.38, which is a quite low correlation rate. This suggests that the models are skillful but in different ways, so the good ensemble method is expected to be able to figure out how to get the best from each model.

I have used a more complex stacking method using Stochastic Gradient Boosting to create the ensemble model. The best achieved accuracy was ~ 79%.

Dataset: whitewine_quality

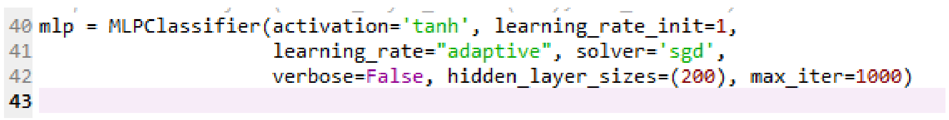
Analysis on dataset

This dataset has 4898 samples and 12 variables, in which the response variable is quality. There is a very big class imbalance: out of 4898 samples but only 20 are of class 3 and only 5 are of class 9. There are not enough samples of those classes to split the data into useable training and test sets and perform cross-validation. However, I will still keep the sample ratio as it is, but there is another option is to group these small classes into one. Another observation is the quality classes are in continuous range, so we can use this dataset as regression or multi classification problem. In this assignment, I considered it as multi classification problem.

Training tools and individual neural networks

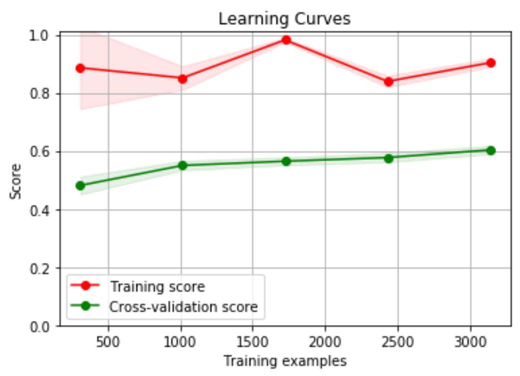
Sklearn library: is a Python opensource library, simple and efficient tool for data analysis, built on NumPy, SciPy and matplotlib. This library has built in classes for MLPClassifier (support multi layer perceptron neural network) which is very flexible for parameter tuning, with different activation functions (relu, tanh, sigmoid), Stochastic Gradient Descent and normal batch gradient descent, adaptive learning rate ...

The best MLP model is trained as following:



```
mlp = MLPClassifier(activation='tanh', learning_rate_init=1,
                    learning_rate="adaptive", solver='sgd',
                    verbose=False, hidden_layer_sizes=(200), max_iter=1000)
```

This model returned train accuracy ~ 100%, test accuracy ~ 62%. Train accuracy is very high in compare to test accuracy is a sign of high variance model. Below is the learning curve of this model with corresponding training examples size.



The MLP model with relu activation function did not give out better accuracy rate (train accuracy ~ 96%, test accuracy ~ 44%), also is high variance.

With the nature of this dataset, it is predictable to have better performance with SVM model. (As I have tried out SVM, the test accuracy of SVM was ~ 67%).

Ensemble model

As pointed out that all individual NN models have high variance problem, to ensemble these NN models, I have use Bagging method to shuffle training samples. But due to the imbalance class problem as mentioned before, this method did not give out much better result. The accuracy rate is about the same as individual NN models.

The Voting method which takes in all models' results and average the predictions of the sub-models when asked to make predictions. This ensemble model even resulted in worse result (test accuracy ~ 60%).

CONCLUSION

In this exercise, I have used the 2 most efficient languages for machine learning and data analysis: Python and R. The languages and libraries are very powerful and flexible in training, tuning, processing data. But along the way, Python and its libraries are somewhat more efficient in training performance, most of the methods are optimized to achieve better training speed when compared with R. R is more suitable for research and academic purposes, since it is easier to learn and setup, but training big dataset on R takes a lot of waiting time.

Ensemble different classifications in most of the case can achieve better results than individual models, since it can make individual models less susceptible to overfitting, and be able to combine good predictions of different models, especially if the individual models have low correlation among each other. In less observable case, however, ensemble model might only achieve similar result as individual models (if correlation rate is high, means that individual models use quite similar skills for prediction).