Optimisation and Evaluation of Classifiers By Max Neil

Introduction

Classifiers are a machine learning algorithm to automatically categorise data in one or more sets of classes. A common example of a classifier is one which can scan through emails to filter them by either spam or not spam (Mesevage, 2020). You can evaluate the performance of a classifier by calculating different metrics. These metrics are: accuracy, confusion matrix, log-loss and AUC-ROC and more (Agrawal, 2023). In this report I will be measuring classifiers based on their accuracy and confusion matrix using the sklearn load_digits() data set. Classifiers can also be optimised to produce a more accurate prediction; this is done by changing the parameter values of the given classifier. I will also be optimising my classifiers to find out the parameters which yield the best results.

I will get the best hyperparameter values by using k fold splitting. I will split the data 10 ways and create a classifier on each split then calculate the mean values of the splits to determine the performance of the parameter values. These parameters will then be used to train a classifier using the whole data set with 85% to train and 15% to test and compare.

My chosen classifiers to optimise and evaluate are random forest, k-nearest neighbours and bagging classifier.

Classifiers

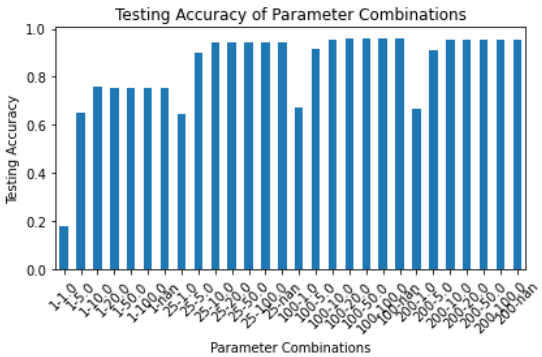
Random Forest Classifier

A random forest classifier is an ensemble of tree classifiers which uses averaging to improve the predictive accuracy of the classifier and control over-fitting (scikit-learn, n.d.). Within the random forest classifier there are some key parameters to consider when tuning these are “n_estimators” and “max_depth”. I have chosen to refine these parameters because of how a random forest classifier works. The number of estimators in random forest refers to the number of individual trees it creates and the max depth refers to how many splits the tree will go before stopping. If this is none then the tree will grow until all leaves are pure or the leaves contain less than the minimum samples split (scikit-learn, n.d.). These are the hyperparameter values chosen:

n_estimators_values = [1, 25, 100, 200]

max_depth_values = [1, 5, 10, 20, 50, 100, None]

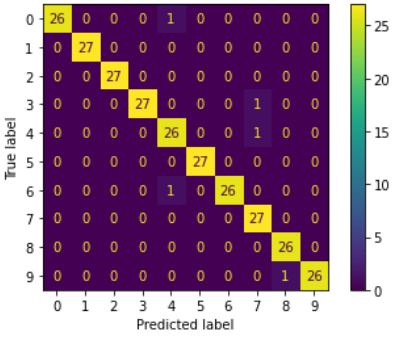
Higher numbers can be chosen to test however when looking at the graphs for these parameters you will see it starts to level out and going higher will not produce any better results.



Having a low estimator value gave worse performance and having an unlimited depth also yielded poor performance. From the tests it was found the best parameter combination was n_estimators = 100 and max_depth = 20. Which gave a mean accuracy of 0.958 however, as shown

by the graph, there are many combinations which gave a similar result.

After getting the information on the best combination of classifiers for random forest I then created a classifier to use on the whole data set. I used the best parameters from the testing to do this.
The accuracy of the classifier on the whole data set was “0.98” which is very accurate. See below for the confusion matrix.



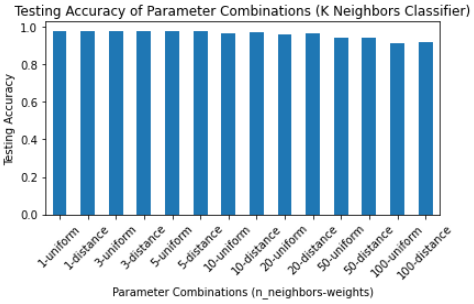
The confusion matrix shows the actual results (y-axis) and the result predicted by the classifier (x-axis). As you can see it made a lot of correct predictions with only a few outliers.

K Neighbours Classifier

“The k-nearest neighbours’ algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point” (IBM, n.d). The hyperparameters I will be tuning are number of neighbours, which changes the number of neighbours, and weights. Weights can have two values, these are ‘uniform’ and ‘distance’. When uniform weights are selected all points in each neighbourhood are weighted equally. If it is set to distance closer neighbours of a query point will have a greater influence than neighbours which are further away (scikit-learn, n.d.). Another parameter I considered was the algorithm. The algorithm parameter can be set to either: ‘ball_tree’, ‘kd_tree’, ‘brute’, ‘auto’ however, when there is no algorithm specified it sets to auto which will choose the best algorithm its self so manually changing these is not necessary. Final parameter values chosen:

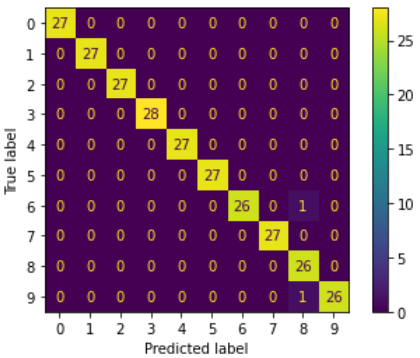
n_neighbors_values = [1, 3, 5, 10, 20, 50, 100]

weights_list = ['uniform', 'distance']



As you can see in the graph changing the parameters doesn’t impact the accuracy of the classifier by much. The highest accuracy was found to be {'n_neighbors': 3, 'weights': 'distance'} which gave an accuracy of 0.978.

With the best parameters found I trained a classifier using the whole data set which gave near perfect results with a final accuracy of 0.993.

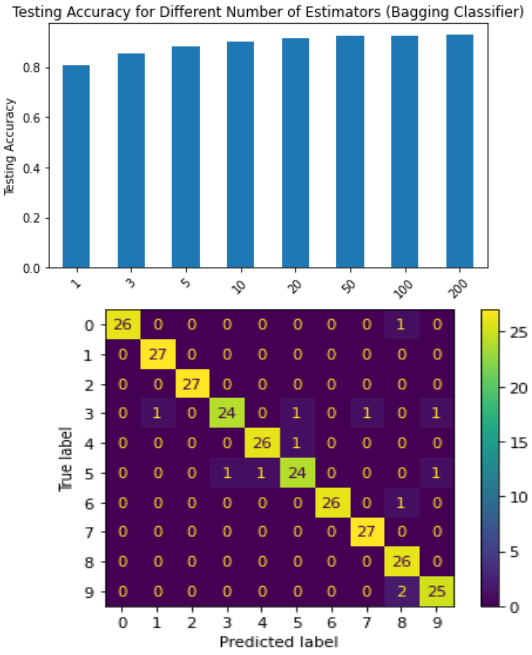


As you can see in the confusion matrix only 2 values were incorrectly predicted.

Bagging Classifier

A bagging classifier is an ensemble of base models which are trained independently in parallel on different subsets of the training data. A final prediction will then be made through majority voting (Dey, 2023). Scikit-learn have a default estimator as a decision tree classifier, this can be changed but for my testing I will leave it as it is. The key parameter that can be refined for a bagging classifier is the number of base estimators in the ensemble, this is define as “n_estimators”. Originally, I had tested values 1 to 500 however this is having a great computational cost and I found that past a certain point the accuracy stays the same so I chose specific values to test instead. These values are:

n_estimators_values = [1, 3, 5, 10, 20, 50, 100, 200]



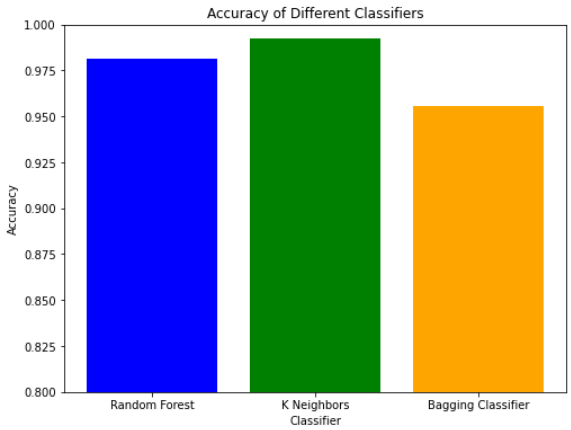
As you can see from the graph the more estimators gave a more accurate result however, past approximately 50 estimators the accuracy balances out and does not change any further. The greatest accuracy was record at {'n_estimators': 200} which gave an accuracy value of 0.928.

Using the parameter values found from testing I trained a bagging classifier on the whole data set and the results were as follows. The accuracy of the classifier was 0.956. See the confusion matrix for the incorrect predictions.

Conclusion

To conclude the research, I have determined which classifier is best on this data set and which parameters should be used to achieve the high accuracy. From testing the three classifiers; random forest, K-nearest neighbors and bagging classifier. The highest accuracy came from K-Nearest neighbors which had an accuracy score of 0.992. The least accurate classifier was the bagging classifier with an accuracy of 0.956.

Classifier	Accuracy
Random Forest	0.981
K-Nearest Neighbors	0.992
Bagging	0.956



References

Mesevage, T G. (2020) ‘Machine Learning Classifiers - The Algorithms & How They Work’, Monkey Learn, 14 December, Available at: https://monkeylearn.com/blog/what-is-a-classifier/ (Accessed: 05 December 2023)

Agrawal, S K. (2023), ‘Metrics to Evaluate your Classification Model to take the right decisions’, Analytics Vidhya, 29 September, Available at: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/ (Accessed: 05 December 2023)

scikit-learn. (n.d.). RandomForestClassifier. Accessed 07 December 2023, available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

IBM. (n.d.). K-nearest neighbors (KNN). Accessed 08 December 2023, Available at: https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.

scikit-learn. (n.d.). KNeighborsClassifier. Accessed 08 December 2023, available at: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Dey, D (2023). ML | Bagging classifier, *Geeks for geeks,* accessed 08 December 2023, available at: https://www.geeksforgeeks.org/ml-bagging-classifier/

scikit-learn. (n.d.). BaggingClassifer. Accessed 08 December 2023, available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html