***Appendix IV.*** *Further description of the hyperparameter optimisation techniques employed and the results of which obtained from the study conducted in Chapter 4.*

*Manual Hyperparameter Optimisation*

An example of the optimisation curves obtained through the hyperparameters explored during the manual exploration of RF can be seen in Figure S6, whilst all other algorithms are depicted within *Appendix III*. Such graphical representations demonstrate the performance change for each model dependent upon the individual varying of a single hyperparameter over a predefined space. Hyperparameter values that reported an increase in performance compared to that of the default values were used in combination for each algorithm. Performance of these optimised models are shown in Table 4.4. The results for manual searching of hyperparameter spaces indicated that with the exception of one case it was able to increase the performance of the models. Improvements to R2 Test are recorded for SVM, KNN, and XGBoost, while RF and DNN additionally demonstrate developments in prediction accuracy, although to a lesser extent. NN was the only exception to this trend, where a slight decrease in performance was observed.

|  |  |
| --- | --- |
| Chart  Description automatically generated with low confidence | Chart, line chart  Description automatically generated |

Figure S6. Optimisation curves of the hyperparameters max\_depth and n\_estimators from the algorithm RF through manual optimisation.

Production of optimisation curves through manual investigation enabled inspection of model performance through visualisation of individual hyperparameters, thus combating generating optimistic and overfitted models. Manually refining hyperparameters is a simple strategy requiring little technical knowledge to implement, while providing researchers insights into the hyperparameter response (Bergstra and Benigo, 2012). However, this approach is not without limitations. Exploration of the local neighbourhood of a given hyperparameter through varying the input and reporting changes in performance is potentially ill conceived. Information gathered through such methods may only suggest how the performance of a sole hyperparameter operates in single instances of the others (i.e., the remaining hyperparameters are unchanged throughout the process), hence ignoring hyperparameters that may work in conjunction (Andonie, 2019). Although strong performance is unexpected from such approaches, improvements upon the default performances were observed, likely due to the selection of inputs for certain hyperparameters that carry a greater weight in model quality. From manual optimisation, hyperparameter spaces that resulted in activity drop-offs were identified, thus informing future ranges to be inputted into both automated approaches, and so reducing computational time.

*Random Search Optimisation*

Random search methodology operates through the selection of random combinations of hyperparameters from a predefined search space. The most significant parameters identified through the randomised search are provided in Table S5, additionally reporting the hyperparameter input space. As shown in Table 4.4, the performance of all algorithms benefited from the random search approach with an increase in performance as compared to manual optimisation. However, RF and KNN both demonstrated minimal improvement, which can be attributed to the low number of hyperparameters that were addressed. Due to the vast hyperparameter spaces that certain algorithms encompass, such as XGBoost and NNs, utilisation of a traditional grid search algorithm (which would usually accompany a sequential manual optimisation) would be unfeasible; hence the selection of randomised search. Bergstra and Bengio (2012) also reported similar results, although focusing only upon NNs, finding that random search can overcome the issues of hyperparameter importance imbalance and exploration of unimportant dimensions that are traditional limitations to manual and grid search optimisations. Random search can therefore be seen as an attractive approach, requiring less time and computational cost, reporting strong results as a result of tuning multiple components during different instances.

˙Table S5. Hyperparameter search dimensions and optimal parameters identified by Random Search. Search dimensions are defined by the start value, which incrementally increases by the step size until reaching the end value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hyperparameter** | **Start** | **End** | **Step size** | **Optimal** |
| *RF* | | | | |
| max\_depth | 10 | 30 | 1 | 30 |
| n\_estimators | 100 | 500 | 10 | 490 |
| *SVM* | | | | |
| Gamma | 0.0012 | 0.003 | 0.00003673 | 0.0012 |
| C | 1 | 10 | 0.47368421 | 8.57894737 |
| Epsilon | 0.001 | 0.02 | 0.001 | 0.018 |
| *KNN* | | | | |
| n\_neighbors | 1 | 15 | 1 | 3 |
| P | 1 | 3 | 1 | 1 |
| *XGBoost* | | | | |
| Eta | 0.1 | 0.15 | 0.01 | 0.1 |
| Gamma | 0 | 0.3 | 0.1 | 0.1 |
| max\_depth | 1 | 8 | 1 | 4 |
| min\_child\_weight | 1 | 10 | 1 | 4 |
| Subsample | 0.8 | 1 | 0.1 | 0.8 |
| colsample\_bytree | 0.5 | 1 | 0.1 | 0.9 |
| n\_estimators | 100 | 250 | 10 | 250 |
| *NN* | | | | |
| Neurons | 50 | 1000 | 50 | 550 |
| dropout\_rate | 0 | 0.5 | 0.1 | 0.2 |
| learn\_rate | 0.0001 | 0.001 | 0.0001 | 0.0003 |
| epochs | 50 | 500 | 50 | 250 |
| batch\_size | 32 | 512 | \*2 | 64 |
| *DNN* | | | | |
| neurons | 50 | 1000 | 50 | 650 |
| dropout\_rate | 0 | 0.5 | 0.1 | 0.3 |
| neurons\_l21 | 50 | 1000 | 50 | 50 |
| dropout\_rate\_l21 | 0 | 0.5 | 0.1 | 0.4 |
| learn\_rate | 0.0001 | 0.001 | 0.0001 | 0.0003 |
| epochs | 50 | 500 | 50 | 500 |
| batch\_size | 32 | 512 | \*2 | 32 |

1l2 refers to the second layer of the network.

*Bayesian Optimisation*

From all the approaches undertaken to obtain optimal hyperparameters, the Bayesian algorithm provided the most significant improvement in performance of the ML models. ML algorithms that contain a large number of hyperparameters, such as XGBoost and NNs, were improved most from the Bayesian algorithm. Meanwhile RF, SVM, and KNN, in which only up to three hyperparameters are being adjusted at a time, reported only marginally improvement, if any, in comparison to simpler approaches. It is expected that the Bayesian method matched, but not exceeded, the improved performance of the results of simpler methods for algorithms with small search spaces, as only a few combinations of hyperparameters may report significant results. Thus Bayesian approaches have most impact during the optimisation of algorithms that require the consideration of large numbers of hyperparameters. The parameter search space and resulting optimal value identified from the Bayesian approach are reported in Table S6.

Table S6. Hyperparameter search dimensions and optimal parameters identified by Bayesian optimisation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hyperparameter** | **Start** | **End** | **Suggest** | **Optimal** |
| *RF* | | | | |
| max\_depth | 10 | 30 | int | 27 |
| n\_estimators | 100 | 500 | int | 499 |
| *SVM* | | | | |
| Gamma | 0.0012 | 0.003 | loguniform | 0.00121 |
| C | 1 | 20 | loguniform | 9.94 |
| Epsilon | 0.001 | 0.02 | loguniform | 0.00852 |
| *KNN* | | | | |
| n\_neighbors | 1 | 15 | int | 3 |
| P | 1 | 3 | int | 1 |
| *XGBoost* | | | | |
| Eta | 0.1 | 0.15 | loguniform | 0.104 |
| Gamma | 0 | 0.3 | uniform | 0.00145 |
| max\_depth | 1 | 9 | int | 5 |
| min\_child\_weight | 1 | 11 | int | 2 |
| Subsample | 0.8 | 1 | uniform | 0.816 |
| colsample\_bytree | 0.5 | 1 | uniform | 0.962 |
| n\_estimators | 100 | 250 | int | 205 |
| *NN* | | | | |
| Neurons | 50 | 1000 | int | 601 |
| dropout\_rate | 0 | 0.5 | float | 0.444 |
| learn\_rate | 0.0001 | 0.001 | loguniform | 0.000376 |
| Epochs | 50 | 500 | int | 236 |
| batch\_size | 10 | 500 | int | 197 |
| *DNN* | | | | |
| Neurons | int | 50 | 1000 | 944 |
| dropout\_rate | float | 0 | 0.5 | 0.161 |
| neurons\_l2 | int | 50 | 1000 | 784 |
| dropout\_rate\_l2 | float | 0 | 0.5 | 0.494 |
| learn\_rate | loguniform | 0.0001 | 0.001 | 0.000321 |
| Epochs | int | 50 | 500 | 498 |
| batch\_size | int | 10 | 500 | 75 |

Bayesian optimisations methods are currently at the forefront of hyperparameter optimisation due to their ability to handle far greater dimensions of parameter space that would be unachievable for simple methods, yielding state-of-the-art results (Hutter et al., 2015). Such results can be attributed to the ability of Bayesian optimisation to utilise knowledge of prior results enabling location, within the search space, of the optimal answer. Computationally, this is achieved through generation of posterior distribution of inputs that best describe the target (statistical value to either be maximised or minimised) which steadily improves as trials progress (Andonie, 2019). Thus, parameter spaces which are worth considering are identified allowing not only strong models to be built, but also having a downstream effect of reducing the number of iterations required. Due to the demand of efficient automatic hyperparameter tuning, a number of Bayesian optimisation software’s have been established with *Hyperopt* (Bergstra et al., 2015) and *Spearmint* (Snoek et al., 2012) being popular examples. However, during the current study the recently developed *Optuna* (Akiba et al., 2019) was selected. The *Optuna* optimisation software can be described as a next-generation hyperparameter optimisation software, improving both searching and pruning strategies, as well as setup procedures. Although, in the context of the present study the most attractive feature can be noted as the define-by-run API. In principle, the define-by-run API enables dynamic construction of the parameter search space, removing the burden from the user; thus, the code is significantly cleaner and easily interpreted. In addition, *Optuna* aids with visualisation of the optimisation process, gaining insight into the dependence of the hyperparameters on the target, as illustrated in Figure S7 and *Appendix III*.

|  |
| --- |
| Chart, scatter chart  Description automatically generated |
| Chart  Description automatically generated with medium confidence |
| Diagram  Description automatically generated |
| Chart, scatter chart  Description automatically generated |
| Chart  Description automatically generated |

Figures S7. Optimisation of the hyperparameters max\_depth and n\_estimators of RF over 50 trials visualised through the *Optuna* software.

Visual assessment of the optimisation procedure that is provided in Figures S7 allows insight into the process. Figures S7a illustrates the optimisation history, where it can be noted that as the number of trials increases scorings observed begin to focus around the optimal. This trend is distinctive to Bayesian optimisation procedures, where, as previously described, the algorithm has been able to learn where the optimal regions are and so can focus efforts there. The focus upon areas of strong predictive performance can additionally be observed in Figures S7b-d. Figures S7b and 4d demonstrate that increasing both the n\_estimators as well as max\_depth produce improved results. Moreover, Figures S7c corroborates the previously discussed theory of Bayesian optimisation where most trials have employed hyperparameters within these ranges. Furthermore, from these visualisations we can learn more about which hyperparameters affect model performance the greatest, with Figures S7e reporting that max\_depth had the greatest impact. As the results of the Bayesian optimisation approach not only reported the strongest performances, but additionally reduced computational time and greater interpretability, the input values identified through this procedure were selected to represent the optimal values for all ML algorithms.

**References (*Appendix IV*)**

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A Next-generation Hyperparameter Opimization Framework. KDD ’19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2623-2631. <http://doi.org/10.1145/3292500.3330701>

Andonie, R., 2019. Hyperparameter optimization in learning systems. Journal of Membrane Computing. 1, 279-291.

Bergstra, J., Bengio, Y., 2012. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research. 13, 281-305.

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D., 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. Comput. Sci. Discov. 8, 014008.

Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical Bayesian Optimization of Machine Learning Algorithms. Advances in Neural Information Processing Systems. 25, 2960, 2968.