Comparison of Heterogeneous and Homogeneous Ensemble-based Predictive Models to Detect Breast Cancer

BS Steyn
(Department of Computer Science)
Stellenbosch University
Stellenbosch, South Africa
21740178@sun.ac.za

***Abstract*—Ensemble-based predictive models utilise a collection of supervised learning models. These collections can either be heterogeneous or homogeneous. The models employed are the k-nearest neighbour(KNN), the Naïve Bayes classifier, support vector machine(SVM), classification tree and shallow neural networks. This paper attempts to implement these classification algorithms and construct an ensemble to test the effectiveness of ensemble-based predictive models. The dataset comprises information related to the identification of malignant and benign breast cancer tumours. The dataset, however, contains quality issues that need to be addressed before the models can be trained. It was determined that heterogeneous ensembles perform the best.**

I. Introduction

No single machine-learning algorithm can be considered the most accurate in all cases. This is due to the effect of an algorithm's inductive bias. This can, however, be mitigated by using a collection of models of the same type, which is referred to as a homogeneous ensemble. This can lead to an increase in the accuracy of the collection, also referred to as an ensemble, by reducing the adverse effect of the individual models' inductive bias [9].

Another approach to solving the inductive bias is to create an ensemble of models of different types; this is referred to as a heterogeneous ensemble. Both of the above ensembles consist of supervised algorithms. Hence, both ensembles require labelled datasets, which are used to train the ensembles to recognise patterns in the data so that the ensembles will be able to identify and recognise these patterns in unclassified data. The quality of the dataset plays a large role in determining how well our ensembles will be able to identify the patterns. Hence, the dataset is first cleaned to deal with missing values, incorrect values, redundant fields and harmful field scales.

Once these data quality problems have been dealt with, this paper attempts to determine which of the models performs the best individually; this model is then used for the construction of the homogeneous ensemble. For the heterogeneous, five different models are used with various voting systems. This paper will provide a comparison between the individual models, the homogeneous ensemble and the heterogeneous ensemble with a majority voting system.

The remainder of this paper is separated into the following sections: Section II provides a background of the various models used. Section III outlines the preprocessing techniques used to enhance the dataset. Section IV provides an overview of the implementation. Section V provides an overview of the empirical procedure used to evaluate the models. Section VI provides the results, which are summarised and concluded in Section VII.

II. Background

This section provides a background on the individual algorithms as well as ensemble-based predictors.

A. K-Nearest Neighbour

The k-nearest neighbour algorithm (KNN), based on the work presented by Evelyn and Hodges [1], is a non-parametric supervised learning algorithm. It is mostly used in classification problems. KNN uses proximity to make classifications or predictions about the grouping of an individual data point. The algorithm functions on the assumption that similar points are positioned in a contiguous space. When attempting to classify an unlabelled query point, the neighbourhood of points around it is considered and subsequently estimated as a member of the majority class. This does, however, mean that the KNN algorithm is extremely sensitive to the distance metric used to define the k nearest points. The value of the constant k influences both the efficiency and the performance of the KNN algorithm. As the value of k the larger the subset of points that need to be considered with each classification. Hence, the larger k, the greater the computational cost of the algorithm. K also has a significant effect on the performance of the algorithm; if k is too large the algorithm any skew in the dataset is exaggerated, causing the majority class to dominate. If k is too small, the algorithm becomes very sensitive to noise. Finding an optimal value for k depends highly on the training set of your data. KNN effectiveness is highly dependent on

the metric used to define the subset of the k nearest points. Popular metrics like Euclidean distance have a bias towards large features in the dataset. To counter this, the dataset should be normalised to achieve the optimal performance of the KNN algorithm.

B. Classification Trees

Decision trees, proposed by Breiman *et al* [5], are a commonly used classification algorithm. Classification trees recursively partition the feature space by solving an optimisation problem which minimises inter-class impurity. This is done by developing a rule at each node to best reduce the diversity of classes on each side. The metric used to measure the reduction in diversity is tantamount to classification success. To reduce the effect of over-fitting, pruning can be performed at the deep nodes.

C. Support Vector Machine

The support vector machine (SVM), based on the work presented by Vladimir Vapnik and Alexei Chervonekis [12], is a supervised machine learning algorithm that classifies data by finding an optimal line or hyperplane that maximises the distance between each class in an N-dimensional space.

D. Naïve Bayes

The naïve Bayes presented by Igor Kononenko [4] is a supervised machine learning algorithm that predicts the probability of an instance belonging to a class with a given set of feature values. Hence, it is a probabilistic classifier. The naive part comes from the fact that the algorithm assumes that all features are independent of one another. This is, however, rarely the case.

E. Neural Networks

Neural networks (NN) are machine learning algorithms that are inspired by the structure of biological neurons. The concept of artificial neurons was first introduced by McCulloch and Pitts [7]. This provided a mathematical model for binary classification tasks. A shallow NN is a neural network that does not contain more than two hidden layers. Backpropagation is one of the most used training methods; it was developed by Rumelhart *et al* [10]. Backpropagation allows the weights of an NN to be efficiently updated. A shallow neural network is generally made of an input layer, an output layer and one or two hidden layers. The size of the input and output layers is set, but the size of the hidden layers can be tuned to find the optimal size for the given problem.

F. Ensemble Learning

The idea of using a collection of models to improve the overall prediction accuracy was presented by Hansen and Salamon [2]. The idea of ensemble learning is to use multiple models, be they homogeneous or heterogeneous, to reduce the inductive bias present in all models. Heterogeneous ensembles also try to receive the benefit of relying on varying models to make up for the weak points of the individual models.

III. Dataset preprocessing

The dataset used in this paper contains various data quality issues. There are missing values, extreme outliers, and redundant fields, and the dataset has an unbalanced class frequency in class occurrences. To truly determine the effectiveness of the different ensembles, the dataset must first have these hindrances removed. As then, the trends in the underlying data will be more visible. The following subsections discuss how these data quality issues were resolved.

A. Missing Data

The dataset used in this paper is beset with "?" to indicate missing values. These "?" values must be dealt with as if they remain; the models may learn to identify patterns in the occurrence frequencies of these values in each of the respective classes (malignant vs benign). The simplest solution would be to drop these values however the dataset is small with regards to the fact that the dataset is small as many of the entries should be kept. A solution to this problem is to replace the missing values ("?") with the mean values of their respective column. This technique was chosen as it is fast to implement. This method is not without its issues. The main issue is that it alters the underlying distribution of the dataset by reducing the variance. However, the true values of the "?" values are impossible to determine, and the number of missing values to be replaced by the mean is small enough not to alter the underlying distribution too severely.

B. Extreme Outliers

The dataset used in this paper contains values that are clearly incorrectly recorded. There are occurrences where values are orders of magnitude larger than the other entries of the same feature. These records are treated as missing values and handled as discussed in the previous subsection. It is important to note that not all outliers are treated in this way; only those that are extreme outliers.

C. Redundant Fields

There is no use in training a classification model on features that convey irrelevant information in relation to predicting whether a tumour is malignant or benign. If these fields are left in the data, they will lead to the models identifying trends in the records of these redundant fields. The models will then make predictions based on these identified trends, which in reality have no relation to which class a tumour belongs to. Examples of these redundant fields from the dataset used in this paper are the *id* and *gender*. Gender is redundant as all the entries are female; hence, this field provides no information. In a similar sense, the *id* which is assigned sequentially by the hospital has no bearing on the malignancy of a tumour.

D. Data Balancing

As mentioned in an earlier section, when there is an imbalance in the proportion of training data associated with a single class, the majority class can begin to dominate the minority class. Some models are more sensitive to this imbalance than

others. However, to achieve the maximum out of all the models used, this imbalance must be addressed. In this paper, the imbalance is addressed by under-sampling the majority class and oversampling the minority class. Oversampling is done by increasing the number of data points associated with the minority class. This is done by randomly duplicating entries in the minority class until the desired number of entries is reached. Under-sampling is when only a subset of data points is taken from the majority class.

E. Normalisation

Some models are very sensitive to large variances in the scale of different features in the dataset. If the variance differs by a large degree, the impact of the larger-scale features will dwarf that of the smaller-scale features. To resolve these issues, the dataset is standardised so that all features have a range of [0,1].

IV. Implementation

This paper implements the discussed NN using the *tensorflow* [6] machine learning library. The rest of the algorithms discussed are implemented using the *Scikit-learn* [8] machine learning library. This library contains implementations for the classification models used in this paper. To visualise the results, the *Seaborn* [13] and *Matplotlib* [3] visualisation libraries are used. The dataset preprocessing is done using the *Pandas* [11] library.

V. Empirical Methodology

This section provides details on how the ensembles and individual methods are first optimised and then evaluated to determine which method performs the most optimally on the dataset used in this paper.

A. Algorithmic Optimization

Before evaluating the success of the respective models and ensembles, they must first be optimised. The main models that are optimised are the KNN algorithm and the NN. For the KNN algorithm, values in the range [1:30] are selected for k. A KNN model is trained using each of these values, and then the trained models are used to evaluate the test dataset. The k value which yields the highest test accuracy is then chosen as the KNN model to be evaluated against and used in the ensembles. For the NN, various hidden layer sizes in the range of [10:20] are selected and then used to configure the NN. The activation function used for the layers is also tested, the three activation functions considered are *relu*, *sigmoid* and *gelu*. The learning rate is also tested for values of [0.0001, 0.001, 0.01]. All possible combinations of these variables are tested, and the combination that produces the highest test accuracy is then chosen.

B. Ensemble Creation

After the optimisation of the algorithms is complete, their optimal configurations, along with the other models discussed in Section II, are run on the test dataset. The module that produces the highest mean test accuracy using a 10-fold method is selected as the model that is used to create the homogeneous ensemble predictor. The heterogeneous ensemble is created using one of each of the models discussed in Section II a majority voting system was used in this paper for both ensembles.

C. Evaluation Metric

To determine if there is any significant statistical difference between the models and ensembles, a 10-fold method is used to collect the test accuracy values of the various models and ensembles. The set of test accuracies for each of the models is then tested for a significant statistical difference using a Friedman test.

VI. Results

The results are presented in the following three subsections. Subsection A provides the outcome for the optimisation of the models. Subsection B provides the outcome of the model selection for the creation of the homogeneous ensemble, and finally, subsection C provides the success of all models and ensembles on the test dataset.

A. Optimal Model Parameters

For the KNN algorithm, the average of 10 runs where the train-test split of the data was randomly performed was used to determine that the optimal performance parameter is $k = 1$ for the dataset used in this paper. Consider the figure:

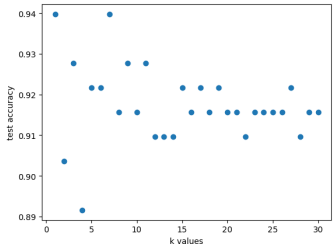
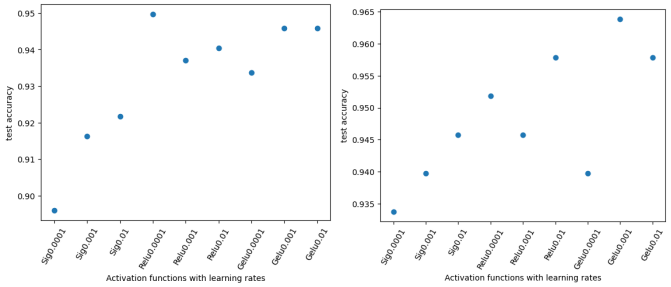


Fig. 1: Each data point represents a pairing of a k value with its average test accuracy

In Figure 1, it can be observed that the k values of 1 and 7 are the values with the highest test accuracy; however, it can be observed that the k value of 1 slightly edges out the k value of 7. Figure 1 also shows that larger values of k tend to have a decreased accuracy, but also that they tend to settle around a test accuracy of 91.5% compared to the optimal k value accuracy, which sits around 94%.

For NN using a similar process of evaluating the mean accuracy over 10 runs, it is determined that the optimal configuration for the dataset used in this paper is *gelu*

activation function with a learning rate of 0.001 and a hidden layer size of 10. Consider the figure below:



(a) Activation functions, learning rate mean pairs

(b) Activation functions, learning rate max pairs

Fig. 2: Two figures showing the pairings of learning rates and activation functions where the values for the various hidden layer sizes have mean summarised using firstly the mean, then followed by the max

From Figure 2, it can be seen that the activation function and learning rate pair of *Relu* with a learning rate of 0.0001 is the most robust when it comes to changes in size, as when the mean is considered, it performs the best. However, when the max values are considered, it can be seen that its performance is rather middle of the range. The pairing that will be used for the remainder of this paper will be the pairing of *gelu* activation function with a learning rate of 0.001.

B. Ensemble Construction

To start with, all the models are tested using the 10-fold algorithm to determine the mean performance of each of the models. The 10-fold algorithm is useful as it allows the algorithms to be trained and tested on the full dataset. This is achieved by retraining the models on every fold as the training and test set shift. Consider Table I:

TABLE I: 10-fold Mean test accuracies

algorithm	mean accuracy after 10-fold	rank
KNN	0.9510064935064936	3
SVM	0.965487012987013	2
Naïve Bayes	0.9012344122603793	5
Classification tree	0.921883116883117	4
NN	0.9691558441558442	1

Table 1 shows that the SVM and NN algorithms have the best performance individually. NN performed the best; however, it will be used in the construction of the homogeneous ensemble. The two ensembles are now constructed.

C. Model Performance on the Test Data Set

Now that the ensembles have been constructed, the performance of all the individual models and ensembles is tested to determine which model to ensemble has the best performance. These tests are done using the 10-fold method, similar as discussed before. It is determined that the best-performing model is the heterogeneous ensemble. However, using the Friedman statistical test with an α value of 0.05, it is determined that there is no statistically significant difference for the top three algorithms. Since the p-value determined using the test is 0.5202774475700467, which is not less than the α value. Consider Table II:

TABLE II: 10-fold Mean test accuracies

algorithm	mean accuracy after 10-fold	rank
KNN	0.9582792207792208	3
SVM	0.9328896103896105	5
Naïve Bayes	0.9012344122603793	7
Classification tree	0.921883116883117	6
NN	0.9529220779220777	4
Homo ensemble	0.9655194805194804	1
Het ensemble	0.9655844155844155	2

The table shows that the models all perform well on the dataset; however, if we compare the two tables, it can be seen that the rankings shift around. However, the fact that the ensembles are first and second makes sense as they utilise the strengths of the algorithms while minimising the weaknesses.

VII. Conclusion

The intention of the paper was to investigate the performance of various methods of ensemble construction against each other as well as the individual algorithms. To conduct a credible comparison, the models were first optimised before they were used in the construction of ensembles. The findings of this paper are that the ensembles yielded a better performance than the individual algorithms, and of the ensembles, the heterogeneous ensemble was found to perform the best. However, the heterogeneous ensemble did not outperform the homogeneous ensemble or the best-performing individual algorithm by a statistically significant degree.

References

[1] Evelyn Fix and J. L. Hodges. “Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”. In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 238–247. ISSN: 03067734, 17515823. URL: http://www.jstor.org/stable/1403797 (visited on 2024-10-24).

[2] L.K. Hansen and P. Salamon. “Neural network ensembles”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.10 (1990), pp. 993–1001. DOI: 10.1109/34.58871.

[3] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[4] Igor Kononenko. “Semi-naive bayesian classifier”. In: *Machine Learning — EWSL-91*. Ed. by Yves Kodratoff. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 206–219. ISBN: 978-3-540-46308-5.

[5] R.A. Olshen Leo Breiman Jerome Friedman and Charles J. Stone. “Classification and Regression Trees”. In: Taylor Francis Group, 1984, p. 368. DOI: https://doi.org/10.1201/9781315139470.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[7] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133. DOI: 10.1007/bf02478259.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Robi Polikar. "Ensemble Learning". In: *Ensemble Machine Learning: Methods and Applications*. Ed. by Cha Zhang and Yunqian Ma. New York, NY: Springer New York, 2012, pp. 1–34. ISBN: 978-1-4419-9326-7. DOI: 10.1007/978-1-4419-9326-7_1. URL: https://doi.org/10.1007/978-1-4419-9326-7_1.

[10] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

[11] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[12] Vladimir Vapnik, Steven E. Golowich, and Alex Smola. "Support vector method for function approximation, regression estimation and signal processing". In: *Proceedings of the 9th International Conference on Neural Information Processing Systems*. NIPS'96. Denver, Colorado: MIT Press, 1996, pp. 281–287.

[13] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.