Ensemble Learning Techniques and A Multi-Arm Bandit Ensemble of Classifiers

36065118
School of Computing and Communications
Lancaster University
Lancaster, UK

***Abstract*—This paper presents a comprehensive review of various techniques of ensemble learning, including Adaboost, Learn++, Stacked Generalization, Mixture of Experts, and Bagging with Borda Aggregation, and compares their performance on a classification task. Furthermore, hybrid ensemble algorithms are introduced along with a novel Multi-Arm bandit based ensemble that divides the feature space among the constituent classifiers. The performance of ensembles is then compared based on the classification accuracy as well as the training and the classification time. The well known Iris data set is used to illustrate the working of the ensembles.**

***Index Terms*—supervised learning, multi-classifier systems, ensemble learning.**

I. Introduction

Ensemble systems are aggregations of multiple classifier systems that were initially developed to reduce variance and increase accuracy in a supervised learning task. However, they have been proven useful in performing several other machine learning tasks that include dealing with missing values, feature selection, dealing with imbalanced data, and confidence estimation, among many others [1]. It is because of their versatility that the ensemble systems enjoy popularity among the machine learning and artificial intelligence communities.

Fundamentally, employing ensemble systems in machine learning tasks is inherently related to the consultation of ”experts” for making decisions in our daily lives. Just like we consult these experts before making a decision on account of their helpfulness in similar matters in the past, ensembles use different classifiers before making a decision in a machine learning task.

When it comes to decision-making, a single perfect classification algorithm does not exist. Each classifier has some error associated with it. In fact, the classification error associated with every classifier has two components: the bias, which is a measure of the correctness of the classifier, and the variance, which is a measure of the uncertainty around the classifier when trained on different samples of the data. The bias and uncertainty of a classifier have a trade-off relationship. This means that, usually, a model with low bias is found to have high variance and vice versa. Thus, depending on the classification result of a single classifier might not always be a good idea. One of the advantages of using ensemble systems is that they can combine the classifiers with relatively low bias in such a way that the outputs of multiple classifiers are averaged out, thereby reducing the overall variance and, thus, resulting in a more accurate model. This is because the constituent classifiers generally agree on the classification of an observation to a label, despite making different errors in the data, causing the overall error in classification to be reduced. The same is illustrated in Figure 1. Along with this, there are other statistical, computational, and representational advantages that ensembles have over single classifiers [2].

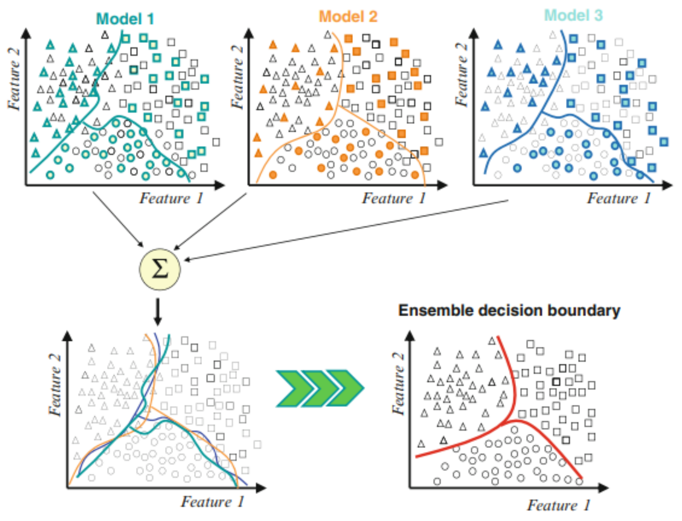

Fig. 1: Smoothing variability using ensembles [1]

In this paper, various ways of combining different classifiers into an ensemble are discussed and their advantages and disadvantages are discussed comprehensively. We also introduce four hybrid ensemble learning techniques: two hybrid ensemble techniques and novel ensembles that divide the feature space among the constituent classifiers using a variation of multi-arm bandit approach based on upper confidence bound. The performance of all the ensembles implemented is evaluated based on their classification accuracy, training time, prediction time and total time. All the ensembles are implemented using the python programming language. For illustrative purposes, all the ensembles are subjected to a binary classification task using the iris data set. The result of the analysis are illustrated in the Discussions section after discussing the methods used for the implementation and analysis in the Methodologies section.

II. Related Work

Arguably, one of the earliest examples of ensemble systems is showcased in [3], where an ensemble is created using a simple linear classifier and a non-parametric classifier. The feature space is divided into two regions using hyperplanes, which can be identified using the SWIFT algorithm [4]. Points falling in one region are learned by the simple classifier, while the others are learned by the non-parametric classifier. These concepts also form the basis of one of the ensembles discussed in this paper. However, one of the most popular works in this research area includes [5], which introduced *boosting* algorithm, showcasing that any weak learner which can perform marginally better than a random guess can be transformed into a strong learner with arbitrarily low error. This also proved to be the foundation for the development of *Adaboost* algorithm [6], which still one of the most extensively used ensemble systems for both classification and regression.

Another simple yet effective ensemble algorithm is *bagging* [7], where different instances of a classifier are trained over bootstrap samples of the dataset. The final prediction is decided either by aggregation or plurality voting, depending on whether the predicted value is a continuous value or a class label.

Various other ensemble learning techniques have been developed, including: a) *random forests* [8] which is an ensemble of decision trees. b) *learn++* [9] which uses ideas similar ideas to adaboost but for dividing the features among different classifiers. c) *consensus aggregation* [10] which uses consensus theory for combining different classifiers. d) *stacked generalization* [11] where a separate classifier is trained on the outputs of the constituent classifiers to aggregate their results. e) *mixture of experts* [12], [13], which is conceptually similar to stacked generalization but instead of training the aggregator on just the outputs of the individual classifiers, original feature space is used as well.

Ensemble systems are now being used in different fields of studies for predictions. For instance, in [14], the authors have discussed several ensemble systems for the problem of data stream classification. In [15], authors suggested a supervised learning based approach to predict the performance of the ensemble system in a use case in real time. This can allow the operator to take remedial measures to change or amend the ensemble system without waiting for the learning/prediction process for the ensemble to end.

III. Methodology

In this paper, ensemble systems including composite classifiers, adaboost, classifier aggregation using Borda voting, learn++, stacked generalization, and mixture of experts are implemented to conduct a study on their performance on a classification problem.

A. Three pillars of ensemble building

To build an efficient ensemble system, diversity among individual classifiers, training individual classifiers, and combining individual classifiers must be considered [1].

- **Diversity among the classifiers:** The individual classifiers in an ensemble need to be diverse in giving predictions. If they always generate similar prediction results, then there is no point in combining them in an ensemble. It has been observed that ensembles perform better if the outputs of the constituents are either uncorrelated or negatively correlated [16], [17]. There are various ways to introduce diversity among classifiers. The classifier algorithms can be different or different instances of the same classification algorithm can be used but with different hyper parameters. The classifiers can be trained on different data samples or on different subsets of features. Thus, various sampling techniques to divide the data space could be employed.
- **Training member classifiers:** The way constituent classifiers are trained in an ensemble is an important aspect of ensemble building. Care needs to be taken that the training strategy employed is not expensive in terms of time and computation. Also, the way the member classifiers are trained could result in noteworthy properties of the ensemble. Popularly used training algorithms include bagging, boosting, stacked generalization, and mixture of experts.
- **Combining member classifiers:** The way outputs of the member classifiers are aggregated is another important aspect of an ensemble. The strategy employed to do so depends on the type of classifiers. For classification tasks, the most commonly used approaches include majority voting, weighted majority voting. Sometimes Borda aggregation is also used as an alternative technique. Another approach is to train an aggregator model to combine the outputs of the constituent classifiers. This approach is employed by stacked generalization and mixture of experts.

B. Composite Classifier Algorithm

This ensemble system draws inspiration from the composite classifier system introduced in [4]. The main idea is to create an ensemble out of the most simple and light-weight classifiers to for most of the prediction and to use a complex non-parametric classifier to predict the data points which cannot be well-predicted by the simple classifier. In order to do so, the feature space is optimally divided into acceptance regions and region of conflict which is achieved by identifying two hyperplanes in the feature space. An illustrative example of the same is showcased in Figure 2. The optimal data space division is when the number of data points in the region of conflict are minimum. The data points which lie in the acceptance region are used to train the simple linear classifier while those which lie in the region of conflict are used to train the complex classifier. During the classification phase, all the points that fall in region of conflict marked by the hyperplanes are predicted by the complex model along with the points where the simple classifier is uncertain (generates a prediction with low probability).

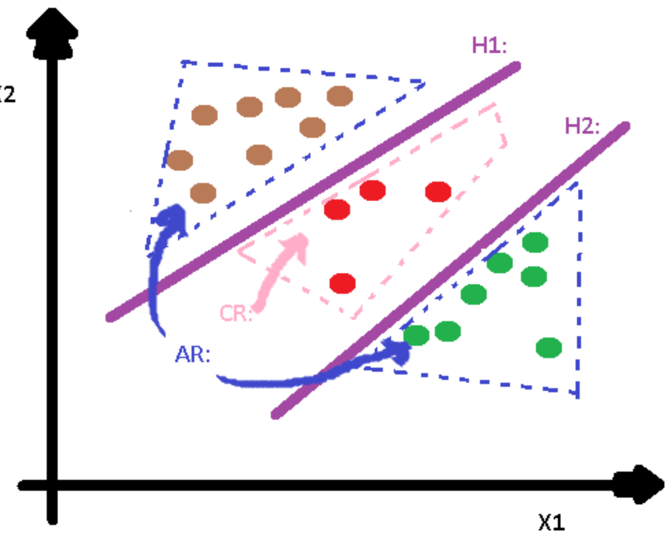

Fig. 2: Division of feature space using two hyperplanes

In order to find the hyperplanes that optimally divide the feature space, an optimization problem is formulated which can be solved using the SWIFT algorithm. The resultant ensemble assumes that the time performance of the model during classification is more meaningful than the time performance of the model during training. Simply put, the time taken during the training phase is much more than the time taken to fit the two individual classifiers, as training involves identification of two hyperplanes and division of data into two regions along with fitting the two models. Since the linear classifier is trained only on the points lying in the region of acceptance, the training accuracy of the model is expected to be 100%. The error in classification for the points in the region of conflict is bound by twice the Bayesian error in the region.

A variation of this algorithm is implemented to conduct the experiment. Instead of formulating a constrained optimization problem, the hyperplanes are identified by using Linear Discrimnant Analysis (LDA) with different hyperparameters. For a binary classification problem, the LDA algorithm outputs a hyperplane. Since the underlying algorithm to fit the hyperplanes is the same, the number of data points falling in the critical region is less. The rest of the algorithm remains the same. Logistic regression and k-nearest neighbors are chosen as the simple linear algorithm and the complex non-parametric model during the implementation.

C. Adaboost

The adaboost algorithm can be seen as an extension of the boosting algorithm. In the beginning of the algorithm, every data sample is assigned is assigned a probability. The data set for the first member classifier is a subset of the training data. The probability for points that are wrongly classified by this classifier are increased. Thus, for the subsequent classifier, the training data set is more likely to contain the data points which were incorrectly classified by the first classifier. In other words, subsequent classifiers are forced to get better in predicting the points that were incorrectly classified by the previous classifiers, which in turn helps in getting better results during aggregation of outputs. In order to handle the case where the classification error of a member classifier is zero, a very small error is introduced in the model during the implementation. The outputs of the member classifiers are combined using weighted voting where the training error corresponding to each classifier acts as the weight against the respective classifier.

The algorithm implemented considers an ensemble of Logistic Regression classifiers each of which are trained on relatively different subset of data. A small variation in the implementation is introduced which forces the member classifier to be retrained in case the classification accuracy of the classifier is less than 50%, to ensure better classification of the ensemble.

D. Combining classifiers with Borda Voting rule

This approach can be seen as an extension of majority voting. Borda voting is useful when we have weighted support for each label given by every classifier instead of just the predicted labels. Thus, this approach takes into account any support that classifiers might have for non-dominant prediction. The idea is to rank order the classes based on the support received. The final output is a vector of class labels ordered by the support against each label.

In the implementation of this approach, classifiers like Logistic Regression, Naive Bayes, Support Vector Machine, and K-nearest neighbours are considered as the constituent classifiers. Prediction probabilities against each class with respect to each classifier is given as the input to the aggregation algorithm.

E. Learn++

Each member classifier is trained on a subset of features. If the data set in consideration has ’f’ features, then only ’n —n ¡ f’ features are considered to train each member classifier. The approach of selecting the n features is similar to the adaboost algorithm. Each feature is initially given a uniform probability. Next, for each classifier, n features are randomly assigned and are used to train the classifier. The probability of the features selected for this classifier is reduced the probabilities associated with other features is increased. The outputs of the classifiers are then combined using plurality voting. In the implementation of this approached, Borda aggregation for combining the outputs of the classifiers has been considered.

A key feature of this algorithm is that it can automatically handle missing data. When aggregating the outputs during plurality voting, the classifiers trained on features consisting of missing values can be simply ignored. Thus, explicit techniques for handling missing data might not be needed.

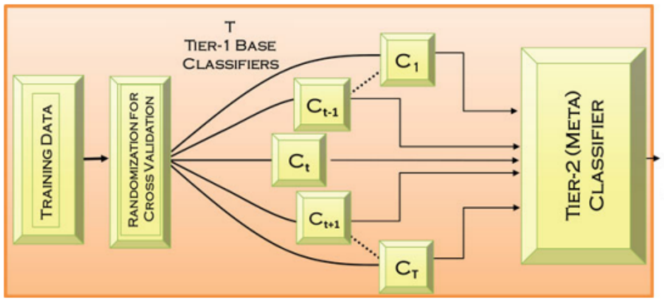

Fig. 3: Stacked Generalization [1]

F. Stacked Generalization

The key feature of this approach is that it uses a supervised learning-based model as an aggregator to combine the outputs of the constituent classifiers. The training data is divided into 'T' equally sized, non-overlapping chunks. For each iteration one of the chunks is used as transient test data for the member classifiers while the rest of the chunks form the transient training data. The outputs the transient testing phase for each of the member classifiers acts as training data for the aggregator. Thus, a total of 'T' iterations are used to realize the data for training the aggregator. The true class label against transient test data is used as the output to be learned by the aggregator. Finally the member classifiers are trained over the entire data for making the actual predictions. An illustration of this approach is given in Figure 3.

G. Mixture of Experts

The underlying algorithm in this approach is similar to stacked generalization especially because this approach also considers a trainable aggregator. The difference is in the data used for training the aggregator includes the actual features as well. An illustration of this approach is given in Figure 4.

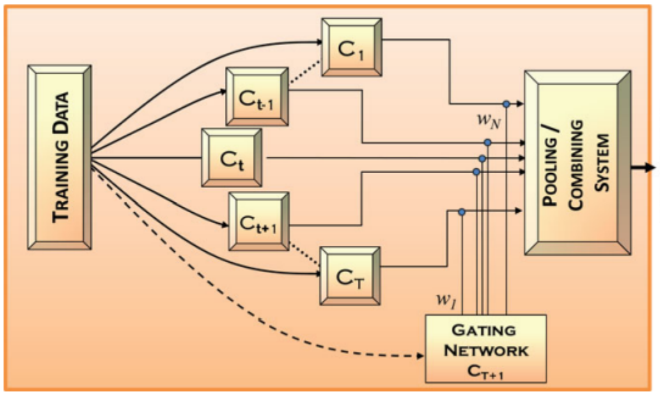

Fig. 4: Mixture of Experts [1]

H. Stacked Generalization with predicted probabilities

This approach is a variation of the stacked generalization approach. It still uses an aggregator to combine the outputs of the member classifiers. However, instead of simply using the predicted result, the predicted probabilities against each class label for each classifier are used as features to train the aggregator.

I. Mixture of Experts with predicted probabilities

This approach is a variation of the Mixture of Experts approach. However, in order to train the aggregator, along with the actual features, the predicted probabilities against each class label corresponding to each classifier are employed.

J. Novel Multi-Arm Bandit based Ensemble

This is the novel approach introduced in this paper. The main idea is to identify a subset of features for each of the member classifiers to be trained. This is achieved by formulating the multi-arm bandit-based problem for selecting the subset of features for the member classifiers in an incremental manner. Thus, for every subsequent classifier, one of the best possible feature subsets is selected, which exploits the features that result in greater accuracy of the model while simultaneously exploring the less used features. An advantage of this approach is that it can be used to identify a subset of features for a classifier, which if used for training will result in the least possible bias.

To find the subset of features for each classifier, we employ the upper confidence bound approach given by the equation 1, where $\overline{X}$ is the mean reward, t is the total number of times any feature was selected, and n_f is the number of time feature f was selected. The reward in this use case is the prediction accuracy for the member classifier.

$$UCE_f = \overline{X} + \sqrt{\frac{2\log(t)}{n_f}} \tag{1}$$

Some amounts of training data are held out for validation in the training phase. The accuracy of the model during the validation forms the reward. The algorithm is designed in such a way that the user can specify the total number of features to be used for training each classifier. Once the member classifiers are trained, the final output is combined using stacked generalisation or Borda aggregation, which means that the flexibility of choosing the aggregation approach is provided. Another added advantage of this approach is that dealing with missing data explicitly is not required. During the output aggregation, the classifiers that were trained over features containing missing values can be ignored. Furthermore, this approach can prove extremely useful in dealing with data with high dimensions.

IV. Results and Analysis

A. The Data set and the Experimental Setup

a) The iris data set: In order to conduct the analysis on the ensembles, we use them for a binary classification task. The data used for the same is the iris data set which is multivariate data set generally used for measurement in taxonomic problems. The data set consists of 50 samples from 3 different species of flowers, namely, iris-setosa, iris-virginica, and iris-versicolor. The measurements of sepal width, sepal length, petal width, and petal length are recorded against each observation. For illustration purposes, only 100 observations corresponding to iris-virginica and iris-versicolor are used to account for the binary classification problem.

The performance of all the ensemble systems is measured in terms of prediction accuracy, training time, prediction time, and total time.

b) Running environment and the data directory: To implement the ensembles, the Python programming language is used. The Python code is hosted on Google colaboratory, a product from Google Research. It provides users with a platform to execute their Python code in a Jupyter notebook style and proves ideal for machine learning and data analysis. It provides users with a Python environment that is preloaded with all the useful libraries and, thus, requires no setup. Moreover, free GPUs are provided to users to perform tasks that require high computation.

Scikit Learn library is used to implement various individual classifiers, while Numpy and Pandas libraries are used for storing and manipulating the data. Matplotlib is used for image visualization.

c) The experiment: The ensembles are subjected to a binary classification problem. The experiment is repeated 10 times for each of the classifiers, i.e.,in total a 100 times. The accuracy of the ensembles, the time required to train the ensemble, the time required to return final predictions for the test data and the total are recorded. The results are illustrated by the figures 5,6,7, and 8, respectively. The formula for calculating the accuracy of model for a classification problem is given by the equation 2, where TP, TN, FP, FN stand for True Positive, True Negative, False Positive, and False Negative, respectively.

$$Accuracy = \frac{(TP + TN)}{TP + TN + FP + FN} \tag{2}$$

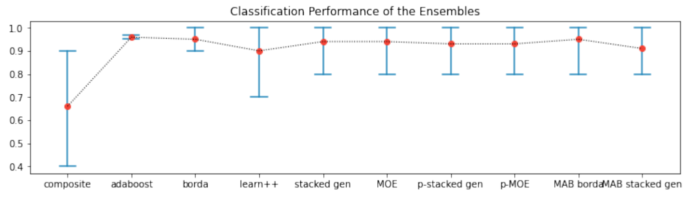

Fig. 5: Classification performance of the ensembles

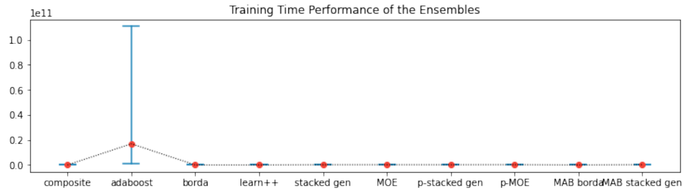

Fig. 6: Performance of the ensembles in terms of training time(measured in nanoseconds)

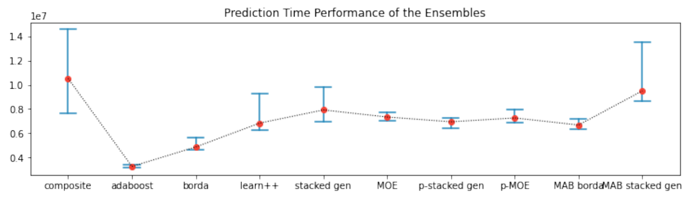

Fig. 7: Performance of the ensembles in terms of prediction time(measured in nanoseconds)

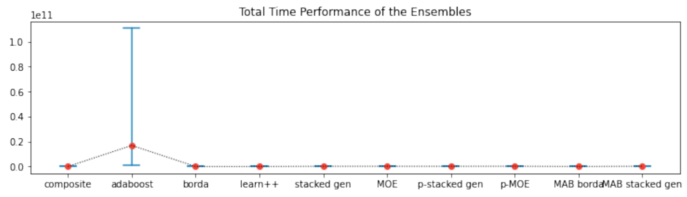

Fig. 8: Performance of the ensembles in terms of total time(measured in nanoseconds)

V. Discussion

A. Accuracy of Prediction

It is observed that all the ensembles perform well when subjected to the binary classification task. Although, a 100% accuracy is never achieved when using adaboost, the prediction accuracy of over 90% is achieved with the slimmest 95% confidence interval. Further, one reason for adaboost to never reach 100% accuracy is because some error was manually introduced in the classifiers so that the probability of a data point being selected for training the subsequent classifiers does not become zero. Not having 100% accuracy can also be justified with the rationale that no classifier is 100% accurate in reality. There can always be an unseen data point which is not correctly classified by the model. Thus, a model is just an estimate of the actual population. It is also observed that combining the constituent classifiers using weighted majority voting with weights being correlated to the classification error helps play in favour of the ensemble.

Although a considerably good classification accuracy is observed for the learn++ ensemble, the interval marked with 95% is relatively large. The low accuracy score could be observed in the case where the feature subsets selected to train the member classifiers generate a low prediction score for most of them. The performance can be improved by considering weighted majority voting like in Adaboost instead of just plurality voting to aggregate the output for the ensemble.

The composite classifier model performs relatively poorly when compared to other ensembles. It is observed that such will be the performance of the composite classifier on relatively small data sets. Although the hyperplanes were to be found in such a way that the region of conflict identified by them had the minimum number of data points, it plays against the performance of the ensemble because, in this way, there is very little data for training the complex member classifier and, thus, it tends to underfit, resulting in poor classification accuracy.

Other ensembles, including the novel ensemble, perform considerably well. The performance of the novel ensemble could also have been observed in the case where weighted majority voting was employed to combine the outputs of the constituent classifiers.

B. Training time performance

The training time for the adaboost algorithm is observed to be substantially higher than other ensembles. This could be especially true for smaller data sets. In the implementation, 60% of the training data is used to train the individual classifiers to account for the diversity among them. For a smaller data set like iris, the majority of these points overlap even when sampled based on probability. In such a case, the prediction results are not changed much for the classifier, and if the accuracy of the classifier is less than 50% the model is trained again and the cycle can continue for a while until a significantly different data sample is learned. The step where the individual classifier is retrained to get better accuracy can be removed to enhance the training time performance, but this could lead to a trade-off between training time and classification accuracy of the ensemble.

It is observed that for the ensembles where a separate classifier is trained as an aggregator, it takes considerably more time to train than the models where Borda voting or majority voting is employed. Somehow, the probability of training a probability-based stacked generalisation requires less training time than the original algorithm.

The novel multi-arm bandit-based approach showcased a relatively low training time with Borda aggregation, indicating effectiveness in time complexity to choose a subset of features for the classifiers.

C. Prediction time performance

For all ensemble algorithms, it is observed that the time taken by the algorithms to generate predictions is considerably less than the time required to train them. The least time with the smallest 95% confidence interval is observed for adaboost, indicating the aggregating classifiers using the simple majority voting or weighted voting rule is the most time-efficient approach. A second-best performance is observed when employing the Borda voting rule. For prediction as well, the probability-based stacked generalisation performs better than the original algorithm. The multi-arm bandit-based ensemble with borda aggregation performs well too.

The Composite classifier system is found to take significantly more time in prediction than other classifiers.

D. Total Time performance

As a majority of the time is consumed in training the ensembles, the total time reflects the same results as that of training time as well.

Overall, it is observed that the novel multi-arm bandit based approach to ensemble making performs significantly well in terms of accuracy, training and prediction times. The performance of the ensemble can be improved further by using weighted voting instead of Borda voting for combining the final outputs. Further, the proposed ensemble brings with it the added advantages of handling missing values and dealing with data with higher dimensions.

VI. Conclusion

In this paper, various aspects of ensemble building, including the importance of diversity among the member classifiers, different ways of training the member classifiers, and aggregating the output of the ensemble, are presented. Different ensemble systems, their key features, and their pros and cons are discussed and evaluated in a comprehensive manner. Three new algorithms, two based on variations in stacked generalisation and MOE, and one based on a novel approach based on a multi-arm bandit approach for feature subset selection, were also proposed. The ensembles discussed are further evaluated by subjecting them to a binary classification problem, and their performance based on accuracy of classification, time taken to fit the model, and time taken to generate predictions is reported.

Further in-depth analysis of the ensembles can be performed by considering more data sets of varying sizes. The proposed algorithm can be improved further by identifying a weighted voting rule to combine the outputs of the constituent classifiers.

References

[1] R. Polikar, "Ensemble learning," in *Ensemble machine learning*. Springer, 2012, pp. 1–34.

[2] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*. Springer, 2000, pp. 1–15.

[3] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979.

[4] B. Sheela and P. Ramamoorthy, "Swift — a new constrained optimization technique," *Computer Methods in Applied Mechanics and Engineering*, vol. 6, no. 3, pp. 309–317, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0045782575900237

[5] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[6] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[7] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[8] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS journal of photogrammetry and remote sensing*, vol. 114, pp. 24–31, 2016.

[9] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[10] J. A. Benediktsson and P. H. Swain, "Consensus theoretic classification methods," *IEEE transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 688–704, 1992.

[11] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[12] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[13] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[14] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, “A survey on ensemble learning for data stream classification,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–36, 2017.

[15] L. Soriano Marcolino, A. S. Lakshminarayanan, V. Nagarajan, and M. Tambe, “Every team deserves a second chance: an extended study on predicting team performance,” *Autonomous Agents and Multi-Agent Systems*, vol. 31, no. 5, pp. 1003–1054, 2017.

[16] A. Chandra and X. Yao, “Evolving hybrid ensembles of learning machines for better generalisation,” *Neurocomputing*, vol. 69, no. 7-9, pp. 686–700, 2006.

[17] Y. Liu and X. Yao, “Ensemble learning via negative correlation,” *Neural networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

Acknowledgements

I would like to thank Dr. Leandro Soriano Marcolino, for introducing me to the concepts of Distributed Artificial Intelligence which in turn peaked my interest in understanding the ensemble systems and their use cases. I would also like to thank my parents for believing in my capabilities and enabling me to pursue higher education. I would like to acknowledge that I have used enhanced my own previous work, i.e., the code which I submitted for the submission of mid-term and end-term assignments for SCC462 module. The code has been significantly modified to handle real data sets. The code is also submitted along with this report.