Ensemble Machine Learning

Pratik Jasani

Johns Hopkins University – Engineering for Professionals

pjasani1@jh.edu

Abstract— The growing world of technology demands complicated tasks with stronger generalization ability. In the modern era, technology keeps getting more and more complicated which requires scientists to keep inventing strategies that meet such demands. In such times ensemble, machine learning can be considered a solution. In some cases, ensemble machine learning can provide robust solutions with better prediction accuracy compared to weak learners. This paper focuses on ensemble machine learning which classifies bagging, boosting, and stacking. It covers the implementation of each type with its advantages and disadvantages.

Keywords— bagging, boosting, stacking, base learner, ensemble machine learning

1. Introduction

As the complexity of the advancing world of technology increases, we are reaching a point where one model fails to reach desired performance/accuracy. To overcome this problem, we can implement ensemble machine learning, which combines multiple models in the prediction process. It helps overcome the technical challenges of a single estimator such as high variance, low accuracy, feature noise, and bias. It is becoming increasingly popular in many competitions like Kaggle and real-world applications [1],[4],[5].

Machine learning is a branch of Artificial Intelligence which attempts to imitate how humans learn using patterns in data. Base learning algorithms like the random forest, regression, neural network, or other machine learning models have limitations and pose a challenge to obtaining higher accuracy, and they also suffer from bias and/or variance. Bias is the difference between the predicted value and the actual value, which is introduced when the model fails to consider the variation in the data. Variance is a measure of how spread out the dataset is. Ensemble machine learning takes the combination of such base learners (weak learners) to make a final prediction.

This paper focuses on different types of ensemble techniques like bagging, boosting, and stacking. It describes how each can be implemented with its advantages and disadvantages. It also compares the ensemble techniques with the performance of a single model (base learner).

1. Ensemble Learning

The ensemble method aggregates the predictions of multiple low-performing classifiers also known as weak learners to make a final prediction. Ensemble methods can be separated into homogeneous and heterogeneous. Heterogeneous methods use an ensemble of the same weak learners while heterogeneous methods use an ensemble of multiple weak learners. The ensemble techniques can be classified into bagging, boosting, and stacking. These techniques help reduce variance, improve the accuracy of the model, and help reduce bias/feature noise [3].

When aggregating the predictions, we consider 3 techniques. The technique used for classification is called max voting which is based on majority voting of predictions. A technique typically used for regression is called averaging which takes the average of all predictions. Sometimes we also use weights of models to produce the final prediction which is called weighted average.

1. Bagging

Bagging is also known as bootstrapped aggregation [6]. It randomly selects subsamples of training data to train each of the base learners in the ensemble. Each base learner is also allowed to select a sample multiple times, meaning the selected subsamples are not guaranteed to be unique. It exploits the independent predictive ability of base learners. All the base learner models are weighted equally when implementing the method.

**Steps of Bagging Ensemble**

1. Create random subsets of pre-processed training data.
2. Create an ensemble of base learners.
3. Randomly select subsamples to train each base learner in the ensemble.
4. Each weak leaner predicts each test datapoint independently.
5. Take the aggregate of predictions to calculate the final prediction.

Diagram

Description automatically generated

Fig. 1 Bagging ensemble steps

As shown in figure 1, bagging can be called the parallel method. It trains each of the base learners simultaneously and independently. Since the outputs of the base learners are averaged, it helps reduce variance in the prediction. Considering further the above steps match the steps required for the Random Forestalgorithm which is one of the most popular variants of bagging ensemble.

1. Boosting

Boosting is a sequential ensemble technique based on regression trees, meaning that the result from a base learner is passed to the next in an effort to further reduce the error [7]. Boosting helps reduce the bias, as each model tried to reduce the errors of the previous model in the sequential chain. Each base learner can have a unique loss function to reduce the error in each sequential model.

**Steps of Boosting Ensemble**

1. Create an ensemble of base learners.
2. Initialize the weight of each data point to 1/n where n is the number of training data points.
3. Train the base learner.
4. Calculate the weighed error rate(e) using the following equation.
5. Calculate the weighted error of the base learner in the ensemble using the following equation. = learning rate
6. Update the weights of incorrectly classified training data points using the following equation.
7. Repeat from step 3 for each base learner in the ensemble.

Diagram

Description automatically generated

Fig. 2 Boosting ensemble steps

To summarize the above steps, each weak learner will train using each data point and update the weights of the weak learner and the incorrectly classified data. The weights of the incorrectly classified data points are weighted higher than correctly classified data. So, the lower weighted data points will be highly considered when making a final prediction.

1. Stacking

Stacking is very similar to bagging, as it also combines various base learners to reduce their biases. Stacking combines multiple machine learning algorithms through meta-learning. Meta-learning is referred to machine learning algorithms that learn from the output of the other machine learning algorithms [8]. Stacking takes the prediction of the trained machine learning algorithms as features to create/train a meta-model.

**Steps of Stacking Ensemble**

1. Create an ensemble of base learners.
2. Train each base learner using training data.
3. Add the predictions to the training data as a new feature.
4. Train the final model using the updated training data.
5. Make the final predictions.

Diagram

Description automatically generated

Fig. 3 Stacking ensemble steps

Meta-model in stacking learns how to best combine the predictions from base learners. It weighs the contributing ensemble members. It allows less weight to be put on the base estimator that does not perform well when predicting. The meta-model can be any machine learning model like simple regression or classification or ensemble learner. An advanced version of stacking uses multiple meta-models to further improve the performance.

1. Database

The dataset being utilized for this paper is the heart attack analysis and prediction dataset. It contains information regarding symptoms that could lead to heart attack such as cholesterol read by BMI sensor, chest pain type, electrocardiographic results, as well as information about the patients. The size of the data set is 303 and the output column indicates whether the person has a high possibility of having a heart attack or not.

1. Pre-processing

The first step of processing is always understanding the data. After importing, understanding, and formatting the data, we would clean the data. The heart failure dataset does not contain any missing values so it can be skipped. The next step of the process would be encoding the categorical data. This step can also be skipped as the dataset does not contain categorical values. Lastly, the data needs to be standardized. To standardize the data scikit-learn StandardScaler was used to convert the data to Z-scores. Figure 4 shows the distribution of the target class after pre-processing.

Chart, bar chart

Description automatically generated

Fig. 4 Heart failure dataset class distribution

1. Machine Learning Model

For this paper, the following machine learning models will be used to create an ensemble. Gaussian naïve bayes, random forest, and support vector classification. These models will be used to create an ensemble for each of the ensemble techniques discussed earlier. The base learners were chosen due to their performance being similar to each other to compare performance with each of the ensemble learning techniques later in the paper. Fig 5 depicts the ROC curve of the chosen base learners.

Chart, line chart

Description automatically generated

Fig. 5 ROC curve of base learners

* 1. Gaussian Naïve Bayes

Naïve Bayes is a probability-based classification algorithm that’s based on Bayes Theorem. It is a family of algorithms that share a “naïve” assumption that every pair of features being classified is independent of each other. Gaussian naïve bayes is a variant of naïve bayes that follows gaussian distribution and supports continuous data. Following default, parameters are used to create a base learner.

**Gaussian Naïve Bayes**

* **var\_something:** 
  1. Random Forest Bayes

Random forest is a commonly used machine learning algorithm that combines the decision of multiple decision trees to reach a final solution. It uses feature randomness to create a forest of independent trees.

**Random Forest**

* **n\_estimators:** 100
* **criterion:** gini
* **min\_samples:** 2
* **min\_samples\_leaf:** 2
* **min\_weight\_fraction\_leaf:** 0.0
* **max\_features:** “auto”
* **bootstrap:** True
* **obb\_score:** False
* **warm\_start:** False
  1. Support Vector Classification

The support vector machine attempts to separate the categories through a hyperplane. In some cases, it is difficult to find a linear relationship between two observations. This is when we can exploit other kernels like polynomial or radial basis function kernel. It outperforms many other techniques in higher dimensional spaces.

**Support Vector Classification**

* **kernel:** ‘linear’
* **degree:** 3
* **gamma:** ‘scale’
* **probability:** True
* **tol:** 1e-3
* **cache\_size = 200**
* **max\_iter: -1**

Chart, bar chart

Description automatically generated

Fig. 6 Base leaner model 10-fold CV accuracies

1. Hyper Parameter Tuning

This section will focus on using scikit-learn GridSeach to evaluate different parameters for each of the ensemble classifiers. For this paper scikit-learn AdaBoost classifier, Bagging classifier, and Stacking classifier will be utilized.

Hyperparameter tuning is determining optimal parameters for a learning algorithm. Machine learning models are susceptible to bias and variance. Choosing the optimal hyperparameters helps avoid over-fitting and under-fitting. A model is over-fitting when the raining error is low, but the testing error is high. A model is under-fitting when the training and the test error are high.

* 1. Bagging Classifier - BaggingClassification

The following parameters were attempted for the bagging classifier for each of the ensemble learners.

* **base\_estimator**: [LinearSVC, RandomForest, GaussianNB]
* **n\_estimators**: [15, 25, 35, 50]
* **max\_samples**: [.8, 1.0]
* **bootstrap**: [True, False]

The base\_estimator parameter represents the base learner being utilized for the bagging classifier. The n\_estimator parameter represents the number of base learners being used to create an ensemble. The max\_samples parameter represents the ratio of training data used to train each base learner in the ensemble. The bootstrap parameter indicates whether a base learner is allowed to be training with a duplicate subset.

**Best parameters:**

* **base\_estimator**: RandomForest
* **n\_estimators**: 50
* **bootstrap**: True
* **max\_samples**: 1.0
  1. Boosting Classifier -AdaBoostClassification

The following parameters were attempted for the boosting classifier to determine the optimal hyperparameters.

* **base\_estimator**: [LinearSVC, RandomForest, GaussianNB]
* **n\_estimators**: [15, 25, 50]
* **learning\_rate**: [0.75, 1.0]

The base\_estimator parameter represents the base learner being utilized for the classification. The n\_estimator classifier represents the number of base learners used to create an ensemble. The learning\_rate hyperparameter represents how fast the model learns from the incorrect classification of previous models.

**Best parameters:**

* **base\_estimator**: RandomForest
* **n\_estimators**: 50
* **learning\_rate**: 1.0
  1. Stacking Classifier – StackingClassification

The following parameters were attempted for the Stacking classifier to determine the optimal hyperparameters.

* **estimator**: [LinearSVC, RandomForest, GaussianNB]
* **final\_estimator**: [LinearSVC, RandomForest, GaussianNB]

The estimator parameter represents different types of base learners being utilized for the algorithm. The final\_estimator hyperparameter represents a type of meta-learner being utilized for the stacking classification algorithm.

**Best parameters:**

* **estimators**: [LinearSVC, RandomForest, GaussianNB]
* **final\_estimator**: GaussianNB

1. Comparisons

The section of the paper compares the base learners with each of the ensemble techniques discussed above to test the accuracy/performance changes. The hyperparameters used for the tests are as described above. The accuracies noted in the tables below were calculated by running a 10-fold cross-validation.

|  |  |  |
| --- | --- | --- |
| **Base leaners** | **Base Learner**  **Accuracy (%)** | **Bagging Accuracy (%)** |
| RandomForest | 81.96 | 84.13 |
| GaussianNB | 80.32 | 81.50 |
| LinearSVC | 83.60 | 83.90 |

TABLE 1  
Bagging vs weak learner Accuracies

Assembling the ensemble of each of the base learners discussed earlier, we can determine that bagging classification slightly improves the accuracy of the base learns. Another approach to further increase the performance could be to create an ensemble of heterogeneous base learners.

TABLE 2  
Boosting vs weak learner Accuracies

|  |  |  |
| --- | --- | --- |
| **Base leaners** | **Base Learner**  **Accuracy (%)** | **Boosting Accuracy (%)** |
| RandomForest | 81.96 | 86.20 |
| GaussianNB | 80.32 | 64.05 |
| LinearSVC | 83.60 | 76.86 |

As we can observe in Table 2, boosting with the random forest as base learners provide better accuracy while boosting with linear support vector classification reduces accuracy.

|  |  |  |
| --- | --- | --- |
| **Base leaners** | **Base Learner**  **Accuracy (%)** | **Stacking Accuracy (%)** |
| RandomForest | 81.96 | 85.24 |
| GaussianNB | 80.32 | 90.16 |
| LinearSVC | 83.60 | 86.88 |

TABLE 3  
Stacking vs weak learner Accuracies

The information in Table 3 was calculated using the random forest, Gaussian Naïve Bayes, and linear support vector classification algorithms as base learners for each test. The adjusted parameter was the meta-learner. For each test, the meta-learner was changed to one of the base learners. To further improve the accuracies there needs to be a deeper analysis using different variations of base learners and meta-learners.

|  |  |
| --- | --- |
| **Base leaners** | **Compute Time(ms)** |
| RandomForest | 204 |
| GaussianNB | 8.01 |
| LinearSVC | 6.66 |

TABLE 4  
Computation time of the base learners

TABLE 5  
Computation time of ensemble classifiers

|  |  |
| --- | --- |
| **Ensemble Classifier** | **Compute Time (ms)** |
| AdaBoostClassifier | 182 |
| StackingClassifier | 837 |
| BaggingClassifier | 5890 |

The values calculated in Table 4 and Table 5 were obtained using the jupyter lab magic command “%%time”. The values for Table 5 were calculated using the optimal hyperparameters obtained in the hyperparameter tunning section. The values for Table 4 were calculated using only the default parameters. What can be deduced from the above results is that using the correct ensemble technique could increase the robustness and performance of any model at the cost of larger computation time.

Ensemble learning allowed us to implement many different variations, but only the most basic variations were utilized for this paper. An example of other variations would be combining multiple base learners for bagging or boosting to determine which combination provides the best performance/accuracy.

1. Advantages/Disadvantages

This section of the paper contains the advantages and the disadvantages of each of the ensemble techniques researched above. This section can help determine which characteristics to look for in the data to best determine the best ensemble techniques to implement.

Bagging significantly reduces variance without an unnecessary increase in bias. If bagging is implemented on a large dataset it can assist in saving computation time by training the model on a small portion of the data while still improving the accuracy.

One of the disadvantages of bagging is the difficulty to interpret what datapoints are being selected for the base learner. There could be cases where some data is not used which results in the loss of important information. It also complicates the interpretability since it uses many individual trees (base learners).

Compared to bagging, boosting is more interpretable. It is robust to outliers and versatile. It helps improve the ability of base models by continuously improving the error from one base learner to the subsequent base learner, which results in higher model performance.

Boosting has a higher cost of modeling time. It is sensitive to outliers, so it is less suitable for datasets with higher noise levels.

An advantage of stacking is that it can harness the abilities of multiple models to improve performance. It also helps reduce bias.

A disadvantage of the stacking is that the computation time is higher because each base learner needs to be trained with all the available training data.

1. Conclusion

This paper presented a comprehensive evaluation of Bagging, Boosting, and Stacking classifiers, with their advantages and disadvantages. Our results demonstrated the following. Bagging classifier helps reduce bias and computation time without increasing bias. Boosting is robust and versatile, as the continuous iteration process helps improve the ability of the base learner models. Stacking allows us to harness the ability of multiple models to improve accuracy. From the research above we can conclude that ensemble learning techniques are simple to understand with a small number of base learners, but the challenge is the requirement of having a deeper understanding of the available data to determine which ensemble technique to implement as the wrong technique could give the worst performance. The above research can tremendously assist in deciding on which techniques to choose. Knowing the basics also allows the user to combine multiple ensemble techniques to improve the performance further. One major limitation of the ensemble method is the explainability, i.e., the knowledge learned by ensemble techniques is not understandable to the user, which makes it harder for decision-makers.

1. Future Work

An interesting area of study would be to investigate if stacked ensemble meta learner can be improved with the use of the neural network, and complex boosting algorithms. Another focus could be to determine what effects combining multiple variations of ensemble techniques provide.

1. Appendix

The experiment was run on an i7 processor with a very low performing GPU, the runtime results might vary from person to person depending on the applications running in the background.

References

1. W. D. Ahmad and A. Abu, “Ensemble Machine Learning Model for Higher Learning Scholarship Award Decisions,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 5, 2020, doi: 10.14569/ijacsa.2020.01105408.
2. Zhang, Jing, et al. “Ensemble Learning from Crowds.” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 8, 1 Aug. 2019, pp. 1506–1519, 10.1109/tkde.2018.2860992. Accessed 14 Mar. 2022.
3. Webb, G.I., and Z. Zheng. “Multistrategy Ensemble Learning: Reducing Error by Combining Ensemble Learning Techniques.” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 8, Aug. 2004, pp. 980–991, 10.1109/tkde.2004.29. Accessed 12 July 2019.
4. P. S. Mung and S. Phyu, "Effective Analytics on Healthcare Big Data Using Ensemble Learning," 2020 IEEE Conference on Computer Applications(ICCA), 2020, pp. 1-4, doi: 10.1109/ICCA49400.2020.9022853.
5. S. Lee, M. Amgad, M. Masoud, R. Subramanian, D. Gutman and L. Cooper, "An Ensemble-based Active Learning for Breast Cancer Classification," 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2019, pp. 2549-2553, doi: 10.1109/BIBM47256.2019.8983317.
6. A. Tartar and A. Akan, "Malignant-benign classification of pulmonary nodules by bagging-decision trees," 2015 Medical Technologies National Conference (TIPTEKNO), 2015, pp. 1-4, doi: 10.1109/TIPTEKNO.2015.7374622.
7. J. Dutta, Y. W. Kim and D. Dominic, "Comparison of Gradient Boosting and Extreme Boosting Ensemble Methods for Webpage Classification," 2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2020, pp. 77-82, doi: 10.1109/ICRCICN50933.2020.9296176.
8. T. Zhang and J. Li, "Credit Risk Control Algorithm Based on Stacking Ensemble Learning," 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA), 2021, pp. 668-670, doi: 10.1109/ICPECA51329.2021.9362514.
9. Z. Zhu, Z. Wang, D. Li, Y. Zhu and W. Du, "Geometric Structural Ensemble Learning for Imbalanced Problems," in IEEE Transactions on Cybernetics, vol. 50, no. 4, pp. 1617-1629, April 2020, doi: 10.1109/TCYB.2018.2877663.
10. R. O. Mounica, V. Soumya, S. Krovvidi, K. S. Chandrika and R. Gayathri, "A Multi Layer Ensemble Learning Framework for Learning Disability Detection in School-Aged Children," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944774.