Comparison of Ensemble methods for Poker hand predictions

Zhiwei Yu	Rajath George Alex	Raghu ram Sattanapalle
New York University	New York University	New York University
zy1129@nyu.edu	rga281@nyu.edu	rrs441@nyu.edu

1. Introduction

We have chosen the UCI Poker Hand data set [1] as our project data. The main goal of this project is to compare the accuracy and performance efficiency of four ensemble methods, namely: Bagging, AdaBoost, Gradient Boosting, and XGBoost to predict the strength of the given hands.

The dataset contains no missing values and Table 1 describes the first 5 samples *(head)* of the training data. Each consecutive column (S1 to S5) denotes the suit color (1: Hearts, 2: Spades, 3: Diamonds, and 4: Clubs) and each consecutive column (C1 to C5) denote the card in the suit (1: Ace, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7, 8: 8, 9: 9, 10: 10, 11: Jack, 12: Queen, and 13: King). The last column is the strength of the poker hand (0: Nothing, 1: 1 Pair, 2: 2 Pairs, 3: 3 of a Kind, 4: Straight, 5: Flush, 6: Full House, 7: 4 of a Kind, 8: Straight Flush, and 9: Royal Flush).

S1	C1	S2	C2	S3	C3	S4	C4	S5	C5	hand
4	9	2	1	2	2	4	7	2	8	0
1	4	3	6	1	12	3	11	2	7	0
1	11	4	1	3	7	4	11	2	1	2
2	9	2	4	3	6	1	9	4	9	3
1	8	2	4	2	11	2	2	2	1	0

Table 1: First 5 samples of the training data

2. Methods and Technical Depth

We compare four ensemble methods: Bagging, AdaBoost, Gradient Boosting and XGBoost - in terms of accuracy and performance efficiency. Accuracy is evaluated using sklearn's *accuracy_score()* method.

Bagging is to grow many deep decision trees, each from a bootstrap resampled training data set, and then to average them to make the final prediction. Unlike other ensemble methods, bagging does not need parameter tuning. It will not overfit when the number of the trees to be averaged becomes larger. We trained this model against the Poker Hand data set and found that this model has a comparable performance to other models *(table 2)* but can be outperformed when the parameters of other models become better tuned *(table 3)*. In this case we changed the maximum depth from the default value of 3 to 10. This makes this method a relatively automatic learning process, in favor of efficiency than accuracy.

Ensemble Methods	Training Accuracy	Testing Accuracy
Bagging	1.0	0.61762
Random Forest	1.0	0.60189
AdaBoost	0.49546	0.48674
Gradient Boosting	0.62409	0.60269
XGBoost	0.59827	0.58097

Table 2: Accuracy scores when parameters for AdaBoost, gradient boosting and XGBoost are set to default

AdaBoost is the predecessor of the more generalized gradient boosting. It was designed specifically for binary classification. Despite its extension to multi-class classification, we found its performance is relatively poor compared to other models.

Gradient boosting is to repeatedly grow decision trees, using gradient descent of the loss function, trying to amend the errors of the previously grown trees, then to combine these weak predictors into a strong one. In gradient boosting, there are three parameters to be tuned, namely the number of trees, the learning rate, and the maximum depth in each individual tree. There is a trade-off between the learning rate and the number of trees. If the training rate is too small, it may take too many trees to fit the training data, thus underfitting. If the training rate is too large, overfitting will occur. The maximum depth in each individual tree controls the model complexity. Although it has a potentially better performance, it takes up more time and more data to tune these parameters.

Ensemble Methods	Training Accuracy	Testing Accuracy
Bagging	1.0	0.61762
Random Forest	1.0	0.60189
AdaBoost	0.90883	0.55111
Gradient Boosting	1.0	0.73611
XGBoost	0.98041	0.67506

Table 3: Accuracy scores when parameters for AdaBoost, gradient boosting and XGBoost have been tuned slightly

XGBoost is an implementation of gradient boosting for better speed and performance. It makes the process of tuning the parameters faster. In further testing, we will compare these models in their running time to evaluate their efficiency and to understand how efficiency benefits from XGBoost's design. We will further also fine tune the parameters for gradient boosting and XGBoost in order to improve accuracy. We will optimize the learning rate, number of estimators and the maximum depth of the tree.

3. Preliminary results

The implementation and data sets used for the analysis can be found at [6]. We generated the confusion matrix and classification reports for all the ensemble methods. In each ensemble method, the results were poor which can be deduced from the poor accuracy results.

The heavy imbalance of the training dataset also contributed to the poor performance of the models. As we can see from table 4-7, the supports for different classes drop dramatically. With most of the samples belonging to class 0 and class 1, which is the nothing and the 1-pair hands, the models have essentially zero correct prediction for other rarer classes. To resolve this issue, we intend to resample the dataset to make each class have more comparable number of samples in further trainings.

Classes	Precision	Recall	f1-score	support
0	0.64	0.80	0.71	3713
1	0.59	0.52	0.55	3224
2	0.48	0.03	0.05	363
3	0.14	0.01	0.01	137
4	0.00	0.00	0.00	30
5	0.00	0.00	0.00	19
6	0.00	0.00	0.00	10
7	0.00	0.00	0.00	3
8	0.00	0.00	0.00	3
9	0.00	0.00	0.00	1

Table 4: Classification report for Bagging

Classes	Precision	Recall	f1-score	support
0	0.61	0.59	0.60	3713
1	0.50	0.60	0.55	3224
2	0.14	0.00	0.01	363
3	0.50	0.01	0.01	137
4	0.00	0.00	0.00	30
5	0.00	0.00	0.00	19
6	0.00	0.00	0.00	10
7	0.00	0.00	0.00	3
8	0.00	0.00	0.00	3
9	0.00	0.00	0.00	1

Table 5: Classification report for AdaBoost

Classes	Precision	Recall	f1-score	support
0	0.77	0.87	0.82	3713
1	0.70	0.70	0.70	3224
2	0.34	0.03	0.06	363
3	0.60	0.04	0.08	137
4	0.33	0.03	0.06	30
5	0.33	0.05	0.09	19
6	0.00	0.00	0.00	10
7	0.00	0.00	0.00	3
8	0.00	0.00	0.00	3
9	0.00	0.00	0.00	1

Table 6: Classification report for Gradient Boosting

Classes	Precision	Recall	f1-score	support
0	0.71	0.82	0.76	3713
1	0.63	0.62	0.63	3224
2	0.21	0.02	0.04	363
3	0.10	0.01	0.01	137
4	0.00	0.00	0.00	30
5	0.00	0.00	0.00	19
6	0.00	0.00	0.00	10
7	0.00	0.00	0.00	3
8	0.00	0.00	0.00	3
9	0.00	0.00	0.00	1

Table 7: Classification report for XGBoost

4. Related Work

There has been a lot of similar work [2][3][4] done in the analysis of various ensemble methods. However, no work till now has focused on the performance and accuracy comparison of conventional ensemble methods such as Bagging, Boosting and Gradient Boosting along with XGBoost [5], which is a recently developed novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning.

5. References

[1] https://www.kaggle.com/c/poker-rule-induction/data

[2] Dietterich T.G. (2000) Ensemble Methods in Machine Learning. In: Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science, vol 1857. Springer, Berlin, Heidelberg

[3] Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning, 36(1/2), 105–139.

[4] Dietterich, T.G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning.

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). ACM, New York, NY, USA, 785-794. DOI: https://doi.org/10.1145/2939672.2939785

[6] https://github.com/rajathalex/Intro-to-Machine-Learning-Project