Group Assignment

The implementation of AdaBoost and comparison with Benchmark Classifiers

Student 1:	480505655
Student 2:	480086811
Student 3:	490234947

Group ID:	Group 64
Unit:	QBUS6850
School:	The University of Sydney

Summary

An AdaBoost model is built and we tune its hyper-parameters using grid search on synthetic dataset. The sklearn AdaBoost and Decision Tree are used to make comparisons with our AdaBoost on a real dataset. We use metrics like score, sensitivity, specificity, precision and auc to compare accuracy and effectiveness. We also compare efficiency, interpretability and model complexity.

AdaBoost has higher accuracy than Decision Tree, although it spends the most time on running. AdaBoost and sklearn AdaBoost are more complex than Decision Tree. Decision Tree, on the other hand, is much easier to interpret than the other two models. We also notice overfitting problems which may cause disadvantages on the model accuracy. Our AdaBoost and sklearn AdaBoost are therefore recommended when making predictions on this real dataset.

Table of content

1.Introduction

In recent decades, attention has increasingly focused on business analysis as means to provide business insight. Business analysis is aiming to reveal the connection of business factors. In the future, business conditions are becoming more complex, leading to an increased demand for reliable and cost-effective analysis tools. Supervised learning is an efficient and popular way to reveal the relationship between factors. A few weeks ago the "AdaBoost" algorithm, later to be referred as the new AdaBoost, was published as a means of supervised learning and its feasibility in the business attracted great attention. This report will first explain the algorithm of Adaboost, and then compares the new AdaBoost with the decision tree and the sklearn version of AdaBoost in terms of complexity, effectiveness, efficiency and interpretability. Finally we will give recommendations for the company based on the research.

2. Algorithm of AdaBoost

Step 1:	Train a number of bootstrapped weak classifiers (decision trees)

Step 2:	Create base estimators

a) Pick the tree with lowest weighted misclassification rate as base estimator

b) Calculate the classifier's voting power

c) Update weights of all data points

d) Using decision functions such as sign() to calculate the predicted values(1 or -1)

Step 3: Stop criteria:

a) If a classifier has zero weighted misclassification rate, choose this classifier and stop.

b) If weak estimators have 0.5 or more weighted misclassification rate, keep the previous classifiers and stops.

c) If the number of base estimators get to the maximum, then stop.

Notes(algorithm):

- The weak classifiers chosen are decision trees. The reason of choosing decision tree is its fast classification, easiness of handling categorical variables and variety;
- Misclassification rate is the sum of errors the weak estimator makes.
- The voting power for the best classifier in each iterative run is calculated as:

$$\alpha_k = \frac{1}{2} log(\frac{1-\varepsilon_k}{\varepsilon_k}) \qquad (1)$$

  where ε_k is misclassification rate;
- The weight of all data points are 1/the number of data points in the beginning;
- The weight for data points in following iteration is calculated as:

$$w^{new} = \frac{1}{2(1-\varepsilon)} w^{old} \text{ if the case is correct} \qquad (2)$$

$$w^{new} = \frac{1}{2\varepsilon} w^{old} \text{ if the case is not correct} \qquad (3)$$

3. Building AdaBoost, evaluation and comparison

3.1 Analysis of all hyper-parameters of AdaBoost on the synthetic

dataset

We are going to tune 9 hyperparameters in AdaBoost including the number of weak classifiers, size of bootstrapped samples trees and the sample size for bootstrapping. We also tune hyperparameters from weak classifiers, such as max_depth to see if varying the parameters of weak classifiers also influence the performance of AdaBoost. In the first step, we use Hold out methods to randomly split the synthetic dataset into a training set and validation set and train the model. Then we use grid search by looking at the test-error vs parameter-value curve to find the optimal value of the hyper-parameter.

3.1.1 Tuning the number of weak classifiers(n_estimators)

n_estimators is the number of base estimators that make up Adaboost. When the number of weak_classifiers increases from 1 to 20, the accuracy of the train reaches to the accuracy just below 1 in a very short time. In terms of validation data, the accuracy reaches its peak of 0.965 when there are 6 base estimators.

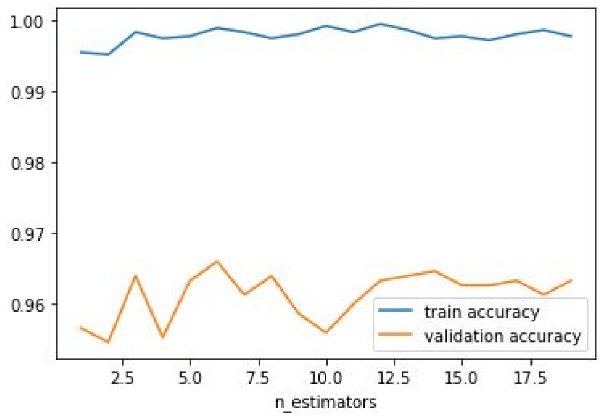


Figure 1: accuracy vs n_estimators

However, validation accuracy does not have an apparent increase, when the number of estimators continue to increase. Thus, we think setting n_estimator to 6 is sufficient for this model to make a good prediction on synthetic dataset.

3.1.2. Analyse the impact of depth of trees(max_depth)

max_depth represents the maximum depth of the trees used to train base estimators. When maximum depth increases, train accuracy increases but validation accuracy decreases.

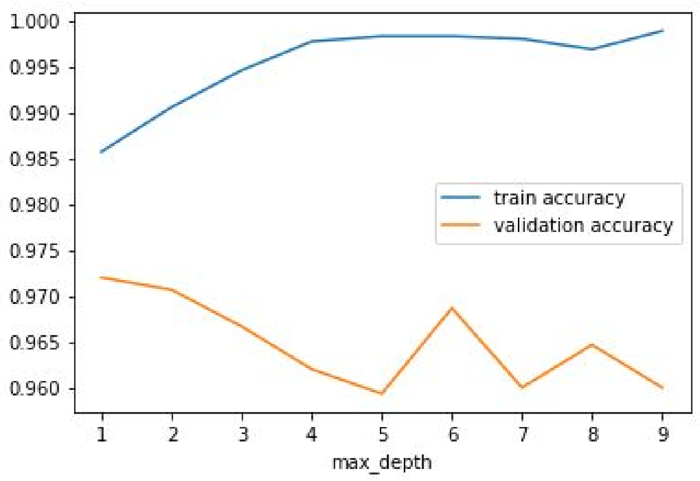


Figure 2: Accuracy vs max depth

The adverse movement of the two lines show an apparent evidence of overfitting. This means increasing the value of max_depth could weaken the generalization of AdaBoost.

The reason for this is that with the increase of depth, the model can catch more information which causes the accuracy to increase, but after max_depth=1, more noise that is not related to target y is also catched by the model. This causes overfitting and thus leads the validation accuracy to decrease.

3.1.3. Analyse the impact of other hyper-parameters

We also consider other 7 hyper-parameters in our analysis. We find that validation accuracy will increase when n_trees, sample_size and max_features increase. Validation accuracy reaches the best result when they are 12, 0.5*X_train and 10 respectively. We also find accuracy decreases when min_saples_leaf, min_weight_fraction_leaf increase.

min_samples_split and max_leaf_nodes have no apparent influence on the model performance.

3.2 Comparison of the implementation of AdaBoost on real dataset

For a business practice, accuracy is always not the only metric that evaluates a model. There are four dimensions that we think are equally important to make an assessment. They are interpretability, model complexity, effectiveness and efficiency.

Interpretability is a question of how easily the models can be interpreted to others and understood.

Model complexity is the degree of difficulty to design and create a model.

Effectiveness is a question concerning how many samples are correctly predicted. It also concerns how many positive and negative samples are correctly classified. It also concerns the overfitting problem which is an extremely important issue.

Efficiency is related to how much time is spent on training the model. In general the less time consumed on model training, the better.

We download the real dataset from Kaggle. The link for this dataset is as follows: https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset.

As there are 17,415 samples in the real dataset, it is sufficient to use the Hold out method to split the real dataset to the training set and validation set. We also need to repeat the tuning process of section 3.1., because we just change our dataset and the optimal value of hyper-parameters may change because of that. However, at this time we only focus on n_estimators and max_depth, because these two parameters seem to have more strong relationships with the performance of AdaBoost when we make analysis on synthetic dataset.

3.2.1 Comparison to the sklearn AdaBoost implementation

(1).Tuning the hyper-parameters of AdaBoost

Graph 9 and 10 shows that when we set n_estimators to be 6, the validation accuracy will reach the peak(0.88). When max_depth=23, the validation accuracy will also be optimized. We assume that AdaBoost has the best performance when we set these two values. We will use this model to make a comparison with sklearn AdaBoost.

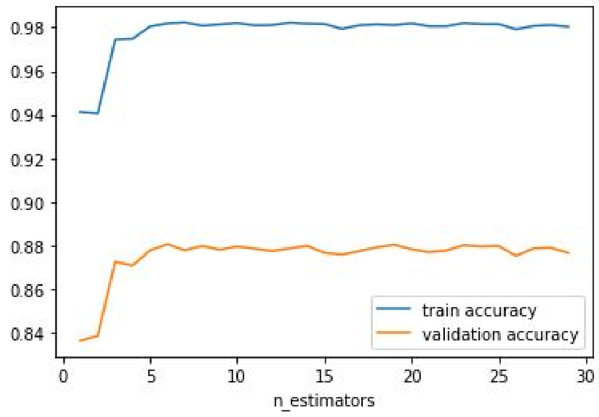


Figure 3: Accuracy vs n_estimators(AdaBoost)

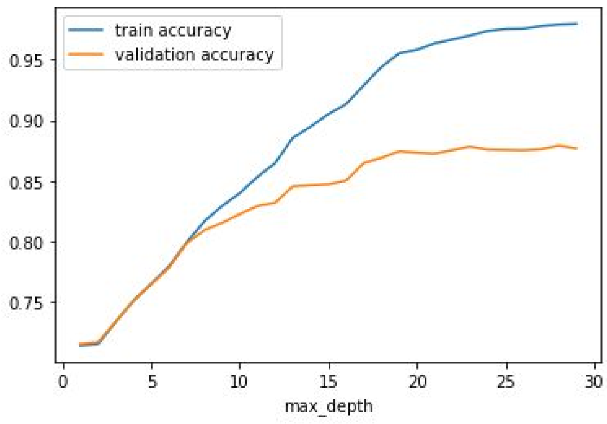


Figure 4: Accuracy vs max_depth(AdaBoost)

(2).Tuning the hyper-parameters for sklearn AdaBoost

From graph 11 we see that the validation accuracy for sklearn AdaBoost increases from 0.85 to 0.93 at n_estimators=30,

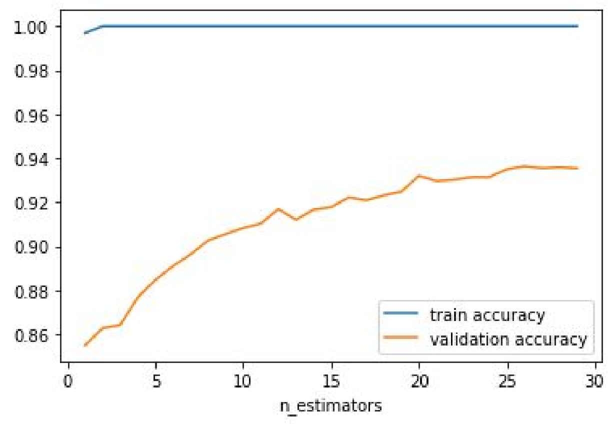


Figure 5: Accuracy vs n_estimators(sklearn AdaBoost)

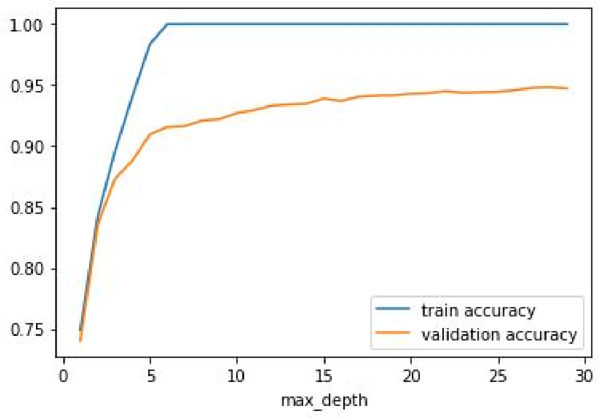


Figure 6: Accuracy vs max_depth(sklearn AdaBoost)

From graph 12 we can see that the accuracy increases and reaches to 100% when max_depth increases from 0 to 5. We see validation accuracy also increase but with slower pace. It reaches its global maximum when max_depth=30.

Thus we use n_estimators=30 and max_depth=(5) as the optimal value for the hyper-parameters for sklearn AdaBoost. We assume that sklearn AdaBoost is optimized when these two values are set.

(3). Make comparison

a. Interpretability

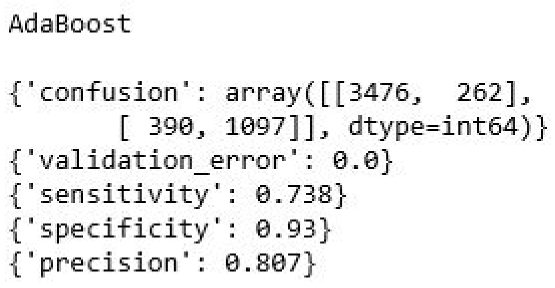
AdaBoost and sklearn AdaBoost are both very difficult to interpret. They are formed from multiple weak classifiers. Sometimes the weak classifiers could be hundreds but people usually do not understand complex models.

b. Model complexity

AdaBoost and sklearn AdaBoost are the same complex when looking at mathematics, because they are built on the same principle. However, sklearn AdaBoost contains more parameters, because it is a mature industrial product and develops for a long time. So in this perspective, sklearn AdaBoost is more complex than our AdaBoost.

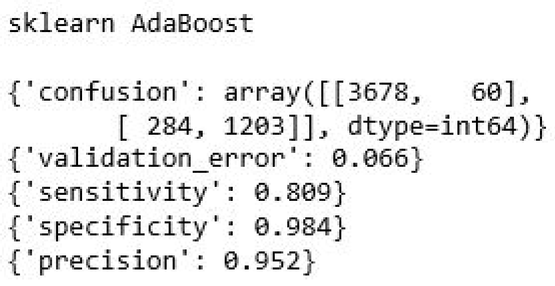
c. Effectiveness

We use metrics accuracy, precision, sensitivity, specificity and precision to evaluate and compare the performance of each model on the real dataset. The picture below illustrates the marks for each model.



```
AdaBoost

{'confusion': array([[3476,  262],
       [ 390, 1097]], dtype=int64)}
{'validation_error': 0.0}
{'sensitivity': 0.738}
{'specificity': 0.93}
{'precision': 0.807}
```

Figure 7: metrics for AdaBoost



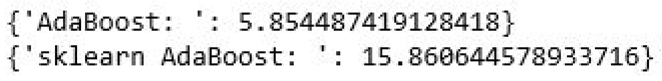
```
sklearn AdaBoost

{'confusion': array([[3678,   60],
       [ 284, 1203]], dtype=int64)}
{'validation_error': 0.066}
{'sensitivity': 0.809}
{'specificity': 0.984}
{'precision': 0.952}
```

Figure 8: metrics for AdaBoost

- Score: The accuracy of our AdaBoost is higher than sklearn AdaBoost.
- Sensitivity: However, the sensitivity of AdaBoost is lower than sklearn AdaBoost. This means more positive samples are found in sklearn AdaBoost. This means sklearn AdaBoost has higher possibility to predict positive samples than our AdaBoost.
- Specificity: The specificity of sklearn AdaBoost is higher than our AdaBoost too. The improvement of the specificity proves that more negative samples are predicted as negative in sklearn AdaBoost. This means sklearn AdaBoost can better help us to find more negative labels
- Precision: sklearn.AdaBoost has higher precision than our AdaBoost. The improvement of precision can prove that sklearn Adaboost really increase the proportion of true positive labels among the whole positive predictions. This means sklearn AdaBoost l really lead to more right positive prediction.

d. Efficiency

We run time.time() to measure how much time is consumed to run these models. We do the measure on the real dataset which has 17,415 rows. The table below shows time spent on each model. We can see that AdaBoost spends 15.85 seconds on running the code, which is 2.7 times slower than AdaBoost. This means our AdaBoost model is much more efficient than sklearn AdaBoost.



```
{'AdaBoost: ': 5.854487419128418}
{'sklearn AdaBoost: ': 15.860644578933716}
```

Figure 9: time spent on running the code

3.2.2 Comparison to sklearn Decision Tree

(1). Tuning the hyper-parameters for sklearn AdaBoost

We tune the hyper-parameters for sklearn Decision Tree using max_depth and min_samples_leaf. In the graph 13 and 14 we can see that when max_depth=15 and min_samples_leaf=10, the decision is optimized. Thus we use this model to make a comparison to our AdaBoost on real dataset.

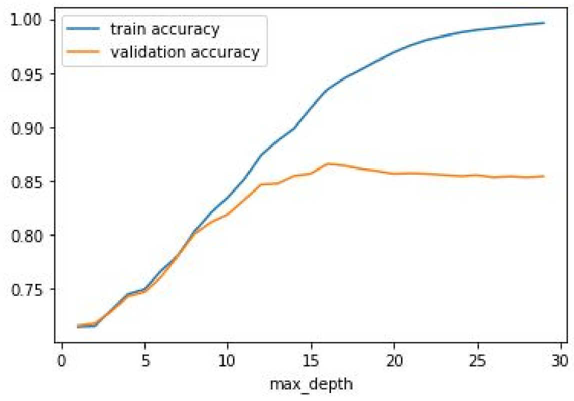


Figure 10: accuracy vs max_depth(Decision Tree)

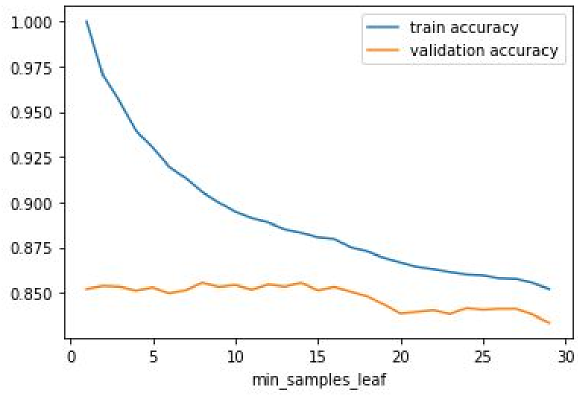


Figure 11: accuracy vs min_samples_leafh(Decision Tree)

(1) Interpretability

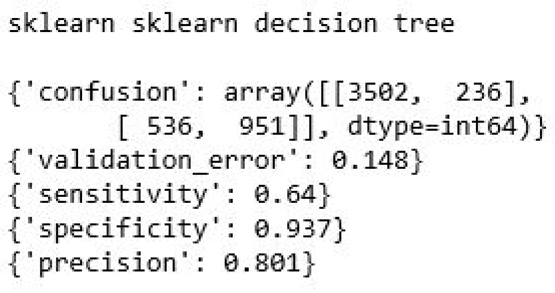
Decision Tree has higher interpretability than AdaBoost. Because it simulates the decision making process of human beings. The mathematics behind AdaBoost is very complicated, so it is very difficult to explain AdaBoost to the public. AdaBoost may also consist of hundreds of weak classifiers to make predictions. This may also make it confusing.

(2) Model complexity

Decision tree is a simpler model than AdaBoost because it is a single model. In comparison, AdaBoost is more complex. It is an ensemble models which owns multiple weak classifiers and its weak classifiers can be any classifiers.

This means users might spend a large amount of time tuning hyperparameters to make it fit to use. On the other hand, the weak classifiers with AdaBoost are related to each other making it even much more complicated.

(3) Effectiveness



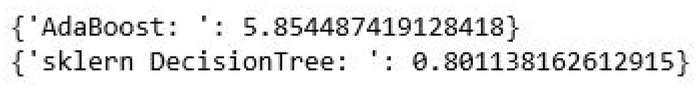
```
sklearn sklearn decision tree

{'confusion': array([[3502,  236],
       [ 536,  951]], dtype=int64)}
{'validation_error': 0.148}
{'sensitivity': 0.64}
{'specificity': 0.937}
{'precision': 0.801}
```

Figure 12: metrics for Decision Tree

- Score: The accuracy of our AdaBoost is higher than sklearn Decision Tree.
- Sensitivity: However, the sensitivity of AdaBoost is also higher than sklearn Decision Tree. This means more positive samples are found when using AdaBoost, which means AdaBoost are better at identifying positive labels.
- Specificity: The specificity of AdaBoost is just slightly lower than sklearn Decision Tree. They have the same ability to classify negative labels.
- Precision: AdaBoost has slightly higher precision than Decision Tree, In this metric, they perform almost the same level.

(4) Efficiency

In the table below, we can see that sklearn Decision Tree has very short running time. It is faster by 7.3 times than AdaBoost.



```
{'AdaBoost: ': 5.854487419128418}
{'sklern DecisionTree: ': 0.801138162612915}
```

Figure 13: time spent on running the code

4. Conclusion and Recommendations

From the comparison between the our AdaBoost classifier, sklearn AdaBoost, and the decision tree, sklearn AdaBoost has the highest effectiveness among the three models. However, sklearn Adaboost spends the most time on making predictions. In comparison, our AdaBoost has the second highest accuracy but the time spent on prediction is smaller than sklearn AdaBoost. On the other hand, Decision Tree has higher interpretability than other two models and it is much simpler than other two as well. All of the models discussed above have their own advantages over other models. My suggestion is that users select a model according to what they need. They do not always need to use the model with the highest accuracy, because a high accuracy and high complexity model may be more time consuming to run than a single model.

References

Dhiraj, K. *Top 5 Advantages and Disadvantages of Decision Tree Algorithm*. 2019. Retrieved From: https://medium.com/@dhiraj8899/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.