Assignment 2-Comparison between Decision Tree, Random Forest, XGBoost

Group members: (Yang Ge (450005028), Haoyue Li (470136555), Suwan Zhu (480090900))

Abstract

In today's data information era, solving multiclass classification problems has always been a key topic in the field of data mining and machine learning. This report will analyze three different tree-based classifiers which are decision tree, random forest and XGboost, and clarify the key principle of these three classifiers. The fashion MNIST dataset is selected to test the performance of these three classifiers. 10-fold cross validation is executed for better analysis as well. The result shows that for accuracy, XGBoost has the highest accuracy, while the running time of XGBoost is obviously much higher than the other two classifiers. We will discuss the relevant concept in detail in this paper.

1 Instruction

In the first assignment, we used the learned classification algorithm to independently write a classifier and classify the images, which helped us to have a deeper understanding of machine learning and a macroscopic view of multi-classification problems.

In this assignment, we have further studied the image classification problem and related algorithms. We learned about three new classification methods which are decision trees, random forests, and xgboost. Since these three methods are all tree-based models, it makes more sense to compare and analyze the data, so we choose these three methods. We use the functions of external open-source libraries scikit-learn and xgboost to build these three models and complete the test of the fashion MINST dataset which is widely recognized as a widely used dataset in the field of image classification research. At the same time, we also constantly adjust the main parameters of the tree model method which is max-depth, finding the optimal depth value and analyze the results.

The importance of this study is to grasp the application scenarios of each classifier by learning the principles of the classifiers of the three tree models and comparing the performance. Through a large number of data analysis and comparison, and constantly adjust the parameters, a more accurate model is built, and finally the prediction results are more accurate. More importantly, with the rapid development of artificial intelligence, the technology of image classification has gradually begun to integrate into people's daily lives. Therefore, a high-performance image classifier can be applied to many fields, making classification work more efficient and intelligent.

2 Previous work

In order to choose appropriate classification algorithms, some researches about classifying similar data sets are referred to. The similar data sets we referred include CIFAR-100, STL-10, SVHN, etc. They are all image data sets and good for developing machine learning and object recognition algorithms. Hundreds of different classification algorithms experimented in various papers show that they work well to classify these images. Common classifiers include SVM, k-NN, Bayes, decision

tree, random forest, XGBoost, neural network, and deep learning. For instance, Kussul and Baidyk (2004) developed a novel neural classifier LImited Receptive Area (LIRA) to train fashion MNIST dataset and the error rate only is 0.61%. Furthermore, a novel method for transfer learning which uses decision forests is presented and it achieves higher accuracy that is 93.78% (Goussies et al., 2014). Similarly, Choi and Song (2018) used a new tree induction algorithm to demonstrate the efficiency of tree model and this algorithm have the accuracy of 91.56% on MNIST dataset.

3 Methods

3.1 Techniques

3.1.1 Decision Tree

Decision tree is a common method of classification and regression. Taking the two-category task as an example, we hope to learn a model from a given training data set to classify the new example. The task of classifying the sample can be regarded as the process of making a decision for the question "Does the current sample belong to a positive class?". Therefore, the production of decision trees is a recursive process. The goal is to create a model that predicts the value of the target variable by learning simple decision rules inferred from the data features.

The key of decision tree learning is how to choose the optimal partitioning attribute. As the partitioning process continues, we hope that the samples contained in the branch nodes of the decision tree belong to the same category as much as possible, that is, the "purity" of the nodes is getting higher and higher. Information entropy is one of the most commonly used indicators for measuring the purity of a sample set. Information entropy is defined as

$$Ent(D) = -\sum_{k=1}^{|y|} p_k log_2 p_k \tag{1}$$

where D represents data set, k represents categories and p_i represents the probability that the k-th category occurs in the entire training dataset. The smaller the value of Ent(D), the higher the purity of the decision tree.

Assuming that the discrete attribute a has V possible values, if a is used to partition the sample set D, then V branch nodes are generated. The v-th branch node contains all samples in D that have a value of a^v on attribute a, which is denoted as D^v. Considering that the number of samples included in different branch nodes is different, the weight of the branch node is given $|D^v|/|D|$, that is, the more the number of samples, the greater the influence of the branch nodes is, so the attribute a can be calculated. The information gain obtained by the sample set D is divided.

$$Gain(D, a) = Ent(D) - \sum_{v+1}^{V} \frac{|D^v|}{|D|} Ent(D^v) \tag{2}$$

In general, the greater the information gain, the greater the "purity increase" obtained by using the attribute a to perform the division.

The information gain criterion has a preference for attributes with a large number of values. To reduce the possible adverse effects of this preference, the decision tree algorithm uses the gain rate to select the optimal partition attribute. The gain rate is defined as

$$Gain\_ratio(D, a) = \frac{Gain(D, a)}{IV(a)} \tag{3}$$

where

$$IV(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} log_2 \frac{|D^|}{D} \tag{4}$$

is called the intrinsic value of the attribute a. The greater the number of possible values of the attribute a (the larger the V), the larger the value of IV(a) is usually.

The Gini index is used to select the partitioning attribute. The purity of data set D can be measured by the Gini value:

$$Gini(D) = \sum_{k=1}^{|y|} \sum_{k' \neq k} p_k p_{k'} \quad (5)$$

$$= 1 - \sum_{k=1}^{|y|} p_k^2 \quad (6)$$

Intuitively, Gini(D) reflects the probability of inconsistent classification of two samples from data set D. Therefore, the smaller the Gini(D), the higher the purity of the data set D. The Gini index of attribute a is defined as

$$Gini\_index(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{D} Gini(D^v) \quad (7)$$

3.1.2 Random Forest

Random Forest, which is also named as Random Decision Tree, is an ensemble and supervised learning method. It can be used for both classification and regression task. It creates a forest randomly and the "forest" is consist of decision trees and merges these trees together to get a more accurate and stable prediction. Therefore, the general idea of the Random Forest method is, to improve the prediction performance of overall result combine learning models through the combination of learning models. The figure below also shows the work process of Random Forest.

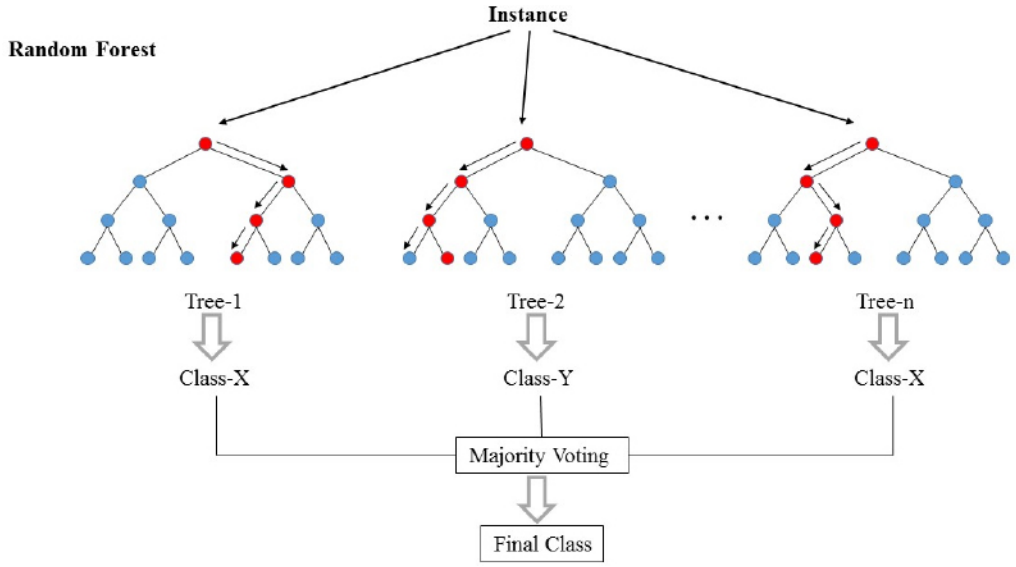

Figure 1: Random Forest

As shown in the figure, it's especially important to choose appropriate numbers of trees also like the depth of each tree in order to achieve an acceptable overall result and avoid overfitting.

Bagging is a typical ensemble technique used in Random Forest model. Each round the algorithm selects n samples from N original samples as a training set, and performs k rounds of extraction to obtain k training sets, the training sets are independent of each other. Then, each time a training set is used to train a model, k training sets get a total of k base models. In a classification task, the prediction result is got through k base models voting, and minority subject to the majority.

3.1.3 XGBoost

XGBoost, which is extreme gradient boosting, is a scalable machine learning system for tree boosting(Chen and Guestrin, 2016). It is available as an open source package. Currently, the

package supports python, R, Java, and Julia. The impact of the system has been widely recognized, it has won so many machine learning competitions.

Boosting is a typical ensemble technique, another technique is bagging, which we have discussed in Random Forest method. While bagging method learn models independently, boosting are learn models sequentially and dependently. We add each new model, which can be learned after each iteration, to previously trained model.

$$F^{(t)} = F^{(t-1)} + f_k^{(t)} \tag{8}$$

where f^t is a weak model. In the end, we would get the final model $F(x)$

$$F(x) = \sum f_k(x) \tag{9}$$

In XGBoost, the base/weak machine learning model is decision tree.

XGBoost is under the Gradient Boosting framework. Gradient boosting combines basis model into one a single stronger learner. Given a supervised classification machine learning problem, we will try to minimize the loss function $L(\phi)$. The gradient boosting method assumes that the loss function can be represented as following:

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_i \Omega(f_k) \tag{10}$$

$l(\hat{y}_i, y_i)$ is a loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i. $\Omega(f_k)$ penalizes the complexity of the model to decrease the effect of over-fitting, which can be represent as

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{i} \omega_j^2 \tag{11}$$

In Gradient boosting method, we would iteratively update the objective function as following

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{12}$$

This means we greedily add the f_t that most improves our model. Then we use second-order approximation in XGBoost to optimize the objective.

$$L^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{13}$$

where

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{14}$$

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{15}$$

are first and second order gradient statistics on the loss function.

The most important part is the scalability of XGBoost in all scenarios.XGBoost provides a parallel tree boosting, and it can run on a major distributed environment, such as Hadoop, and can solve problems beyond billions of examples.

The most time-consuming part of decision tree learning is to get the data into sorted order to decide how to split the data in each decision node. XGBoost store the data in in-memory units, which is called block to reduce the cost of sorting. In block, data is stored in the compressed column format sorted by the feature value. Collecting statistics for each column can be parallelized, this can give us a parallel algorithm for split finding. Then cache-aware pattern also helps to solve the problem about indirect fetches of gradient statistics by row index. Block Compression and block sharding techniques are used to improve the out-of-core computation, which is to utilize disk space to handle data that does not fit into main memory.

Cache access patterns, data compression, and sharding are essential elements for building the scalable end-to-end system for tree boosting.

3.2 Techniques Comparison

In this experiment, we aim to compare the performance of these three tree model in a classifier task. As they are a different form of tree model there are also some differences among the three models' mathematical theory. The simplest model, Decision Tree algorithm is consist of one tree, which means, when making the prediction the result is just decided by one tree. In contrast, Random Forest algorithm is consist of many decision trees and aimed to achieve a more accurate and stable prediction through merging these decision trees together. It likes bagging decision tree learn models independently and gets the final result by many decision trees' vote. This is where XGBoost is different form. Although Random Forest and XGBoost are both ensemble technique, the XGBoost is the learning model which works sequentially and dependently while trees in Random Forest model are works independently. Therefore, we guess that, compared with Decision Tree, Random Forest and XGBoost would have a better accuracy but cost more as they are more complicated. The performance of these three models will be shown in the experiment results.

3.3 Pre-processing

In this experiment, the classification methods we used are decision tree, random forest, and xgboost, they are both tree-based classifier, and they are both kind of scale-invariant. It does not matter on which scale the features are. The only pre-processing we did is standardization.

Standardization. In the field of machine learning, different evaluation indicators often have different dimensions and dimension units. Such situations will affect the results of data analysis. Feature standardization can make the value of mean for each feature in dataset equal to 0, and make variance equal to 1. It is a common requirement for many machine learning method. The general method of applying standardization is that we first determine the mean and standard deviation for each feature, then subtract the mean from each feature, in the end, we divide the values by corresponding standard deviation.

$$x' = \frac{x - \bar{x}}{\delta} \tag{16}$$

Where x is the original feature vector, $\bar{x}$ is the mean, δ is the standard deviation correspondingly.

3.4 Design choices

Model and its complexity. We choose Decision Tree, Random Forest, and XGBoost as the classifiers of this assignment. The reason is that they are very related as they are both tree-based models. Random Forest and XGBoost are two different ensemble learning methods, which can use one classifier as the base, and make the base/weaker classifier stronger. For simplicity, we just adjusted max_depth parameters in all three methods and the number of trees in Random Forest method to adjust the complexity of the models.

Programming language and existing libraries. We used python3 in this experiment. The libraries we used are scikit-learn for Decision Tree and Random Forest, and xgboost library for XGBoost,

Computer infrastructure. As Decision Tree and Random Forest in scikit-learn library do not have many speeding-up options, and they are already very fast, we did not do much speeding-up on these two. For XGBoost, we tried different tree_method, which are exact, approx and gpu_exact to see the runtime performance.

4 Experiments and Discussion

Cross validation. If we simply use all training data to train our model, we could get a model that performs well on training data but may perform poorly on test data. What we want is to get a model that has good performance in general data. Cross-validation is a method that can assess how well a model generalizes by training on a portion of the instances and testing on the remaining.

We use 10-fold cross-validation in our experiments. The idea is to divide the training data

into 10 sections, and then train and evaluate the classifier 10 times, each time training on 9 of them and leaving aside a different section for evaluation. By doing this on different parameters and comparing the results, we then can choose an appropriate parameter for training.

4.1 Decision tree

In this study, the algorithm implementation of the decision tree is based on scikit-learn library to achieve decision classification. We will interpret the parameters of the decision tree classification module DecisionTreeClassifier and optimize the parameters to analyze the results.

- Max_depth. The maximum depth of the decision tree can be left undefined by default. If not entered, the decision tree does not limit the depth of the subtree when creating the subtree. In general, this value can be ignored when there is little data or features. If the model sample size is large and there are many features, it is recommended to limit this maximum depth. The specific value depends on the distribution of the data.

We first load the training and the testing data, doing pre-process on both of them. We use 10-fold cross-validation to train models on each max depth parameters. Then we choose max_depth parameter with the highest accuracy score and use it to train model and predict test data.

Max depth influence. We measure the time and accuracy with different max depth value to explore appropriate model complexity when training decision tree model. The result is shown below:

max_depth	Accuracy(%)	Total Time(s)
2	35.67	37.39
4	65.15	72.57
6	73.26	103.72
10	81.27	170.1
15	81.63	236.3

Table 1: Cross Validation Time and Accuracy value with different max_depth in DT

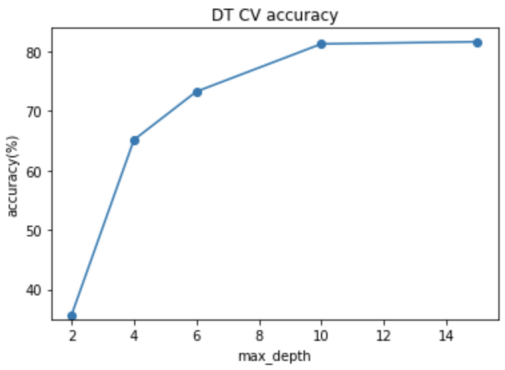

Figure 2: Decision tree cross validation average accuracy

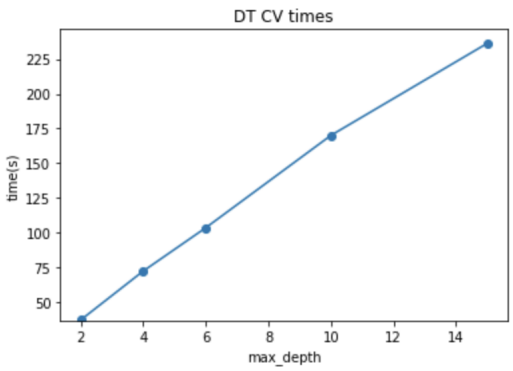

Figure 3: Decision tree cross validation time

When we adjust the max_depth parameter, we found that with the max depth of the tree increase from 2 to 10, the accuracy grows up significantly. When the depth is 10 and 15, the accuracy is not much different. The highest accuracy is 81.63% and can be obtained when the max depth value is 15.

As the maximum depth value increases, the running time for each iteration also increases. From the experimental results, the two are almost linear.

Therefore, we choose max depth 15 for the following analysis, as we got the best accuracy rate with it and the time for each iteration is also acceptable.

Train model with max depth 15. Then we use all training data and parameter max_depth 15 to train our decision tree model and predict. The accuracy is 80.94%, the training time is 26.36 seconds.

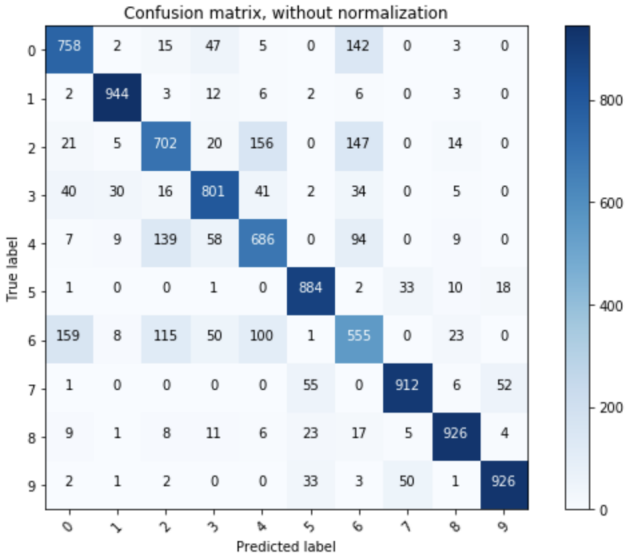

Figure 4: Confusion matrix for max_depth 15

Label(L)	Precision	Recall	F-score
0	0.78	0.76	0.77
1	0.97	0.94	0.95
2	0.66	0.7	0.68
3	0.83	0.8	0.81
4	0.68	0.69	0.69
5	0.93	0.88	0.91
6	0.55	0.56	0.55
7	0.89	0.91	0.9
8	0.92	0.93	0.92
9	0.91	0.93	0.92

Table 2: Precision, recall, f1 score for max_depth 15

From the confusion matrix figure we can see that the number of correctly predicting label 6 is obviously lower than other labels. In decision tree model, it is easy to predict as label 0, 2, 4 instead of label 6. For label 6, precision, recall and f1 score is much lower than other labels.

4.2 Random Forest

In this part, we implement Random Forest classifier based on scikit-learn library. In order to make the comparison more scientific, all parameters in the Random Forest model are same as them in Decision Tree which we implementing before except the numbers of trees we chose. Therefore, the two variables in this model are max depth and the number of trees.

4.2.1 Number of trees

Firstly, we test the accuracy with different Number of trees and Max depth. We found that with a fixed Max depth, the parameter Number of trees just causes a slight difference in the accuracy as the Figure 2 shows. Therefore, we chose a scientific value of the Number of trees which is 100.

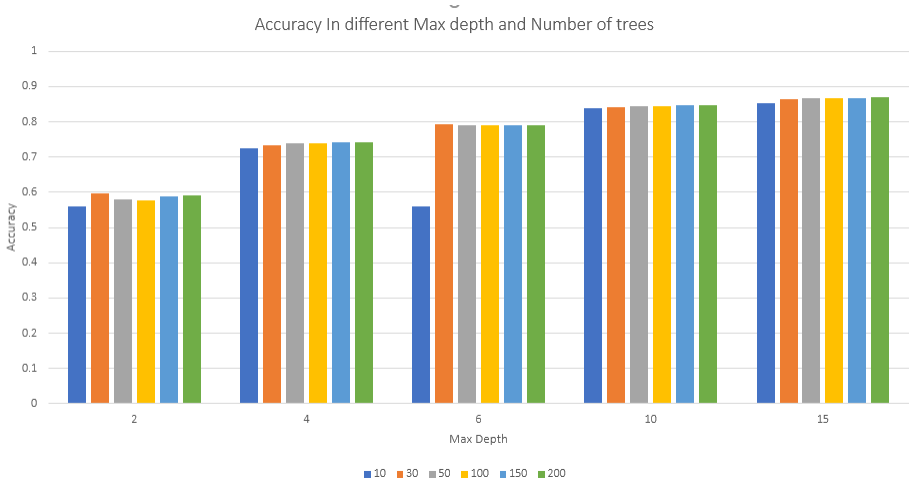

Figure 5: Accuracy-Number of Trees

4.2.2 Max Depth

Secondly, with the fixed Number of tree 100, we change the Max Depth to explore the changes in the running time and accuracy. 10-fold cross validation is also implemented in this step. The result is shown below.

max_depth	Accuracy(%)	Total Time(s)
2	58.67	88.6
4	74.86	163.3
6	80.03	235.0
10	85.41	389.1
15	87.68	541.2

Table 3: Time and Accuracy value with different max_depth in RF

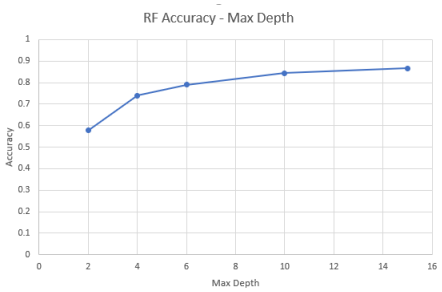

Figure 6: RF accuracy

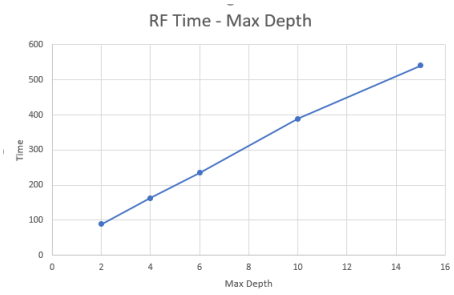

Figure 7: RF cross validation time

It clearly shows that as the increase of Max Depth, the accuracy becomes higher and the running time becomes longer. In this experiment, when Max Depth is 15 we got the highest accuracy which is 87.68 and training time is 61.52 seconds. The details are shown below.

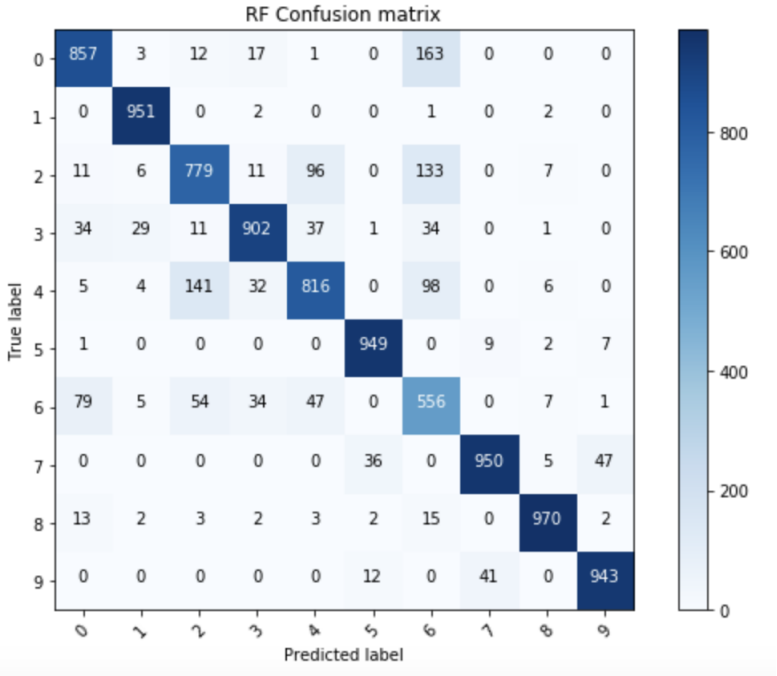

Figure 8: RF Confusion matrix for max_depth 15

Label(L)	Precision	Recall	F-score
0	0.81	0.86	0.83
1	0.99	0.95	0.97
2	0.75	0.78	0.76
3	0.86	0.90	0.88
4	0.74	0.82	0.78
5	0.98	0.95	0.96
6	0.71	0.56	0.62
7	0.92	0.95	0.93
8	0.96	0.97	0.96
9	0.95	0.94	0.94

Table 4: RF Precision, recall, f1 score for max_depth 15

For the confusion matrix figure, if the diagonal cell is dark blue, it means the prediction for thecorresponding label is accurate, the darker, the more accurate. For label 6, f1 score is much lowerthan the others, precision and recall are both relatively low, especially for the recall. The low recallfor label 6 means that the model is very likely to predict the data, which have the true label 6, toother labels

4.3 XGBoost

We used XGBoost classifier in python XGBoost library. It has many parameters, in this experiment, we just focused on one of them.

- max_depth. Maximum depth of a tree. The default value is 6. 0 indicates no limit. Increasing this value will make the model more complex and more likely to overfit.

This parameter would affect the complexity of the model, the appropriate parameter would make model accurate, otherwise model could be underfitting or overfitting. Another parameter called num_boost_round , which is the number of trees, could also affect the complexity of the

model. Fortunately, XGBoost library in python provides early_stopping_rounds parameter, which means training processing would automatically stop if validation error does not decrease on early_stopping_rounds round(s).

We first load the training and the testing data, do pre-process on both of them. We use 10-fold cross-validation to train models on each max depth parameters. Then we choose max depth parameter with the highest accuracy score and use it to train model and predict test data.

Max depth influence. We measure the time and accuracy with different max depth value to explore appropriate model complexity when training XGBoost model. The result is shown below:

max_depth	Accuracy(%)	Total Time(s)
2	86.4	1010.4
4	88.96	1923.4
6	89.04	2218.3
10	88.96	3307.1
15	88.81	6052.6

Table 5: Cross Validation Time and Accuracy value with different max_depth

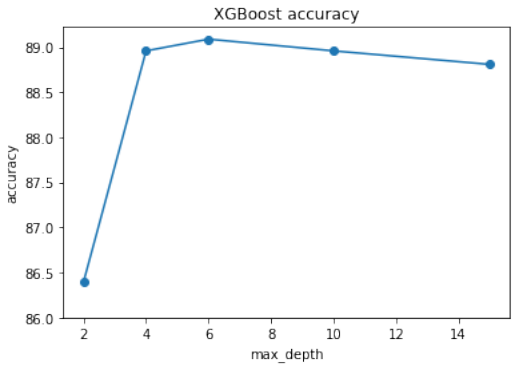

Figure 9: XGBoost cross validation average accuracy

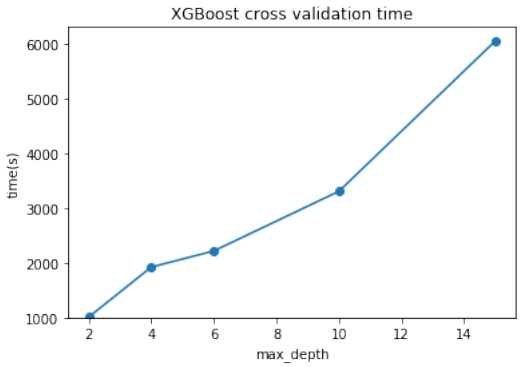

Figure 10: XGBoost cross validation time

For accuracy, the highest accuracy is 89.04% and can be obtained when the max depth value is 6. As the max depth value increase, the accuracy has been increasing until the max depth value is equal to 6, after that, it turns to decreases.

For time for each iteration, as expected, as the max depth value increases, the time for each iteration is also increases.

We then choose max depth 6 for the following analysis, as we got the best accuracy rate with it and the time for each iteration is also acceptable.

Train model with max depth 6. Then we use all training data and parameter max_depth 6 to train our model and predict. The accuracy is 89.09%, the training time is 211.3 seconds.

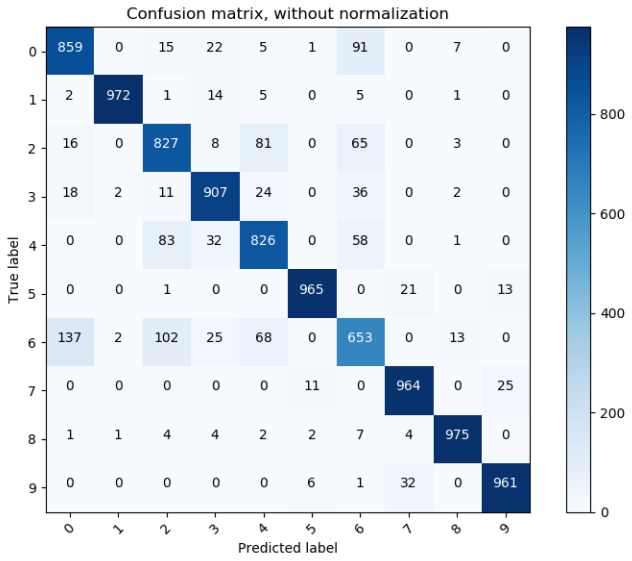

Figure 11: Confusion matrix for max_depth 6

Label(L)	Precision	Recall	F-score
0	0.832	0.845	0.859
1	0.995	0.983	0.972
2	0.792	0.809	0.827
3	0.896	0.902	0.907
4	0.817	0.821	0.826
5	0.980	0.972	0.965
6	0.713	0.682	0.653
7	0.944	0.954	0.964
8	0.973	0.974	0.975
9	0.962	0.961	0.961

Table 6: Precision, recall, f1 score for max_depth 6

For the confusion matrix figure, if the diagonal cell is dark blue, it means the prediction for the corresponding label is accurate, the darker, the more accurate. For label 6, f1 score is much lower than the others, precision and recall are both relatively low, especially for the recall. The low f1 score for label 6 means that the model is very likely to wrongly predict the data.

4.4 Result Comparison

In this step, we put the performance of Decision Tree, Random forest and XGBoost together to make a comparison in the aspect of accuracy and running time. For Random Forest and XGBoost the number of trees are fixed in 100.

Accuracy. Among these three models, with any Given Max depth, the accuracy of XGBoost is highest while the decision tree is lowest. It's obviously that XGBoost works better in the aspect of prediction accuracy. The details are shown below.

Running Time. Among these three models, with any Given Max depth, the running time of Decision Tree is least while the XGBoost is longest. And the running time of XGBoost is much longer than the other two model. The detail is shown below.

Figure 12: Accuracy of Three Models

Figure 13: Running Time of Three Models

4.5 Personal Reflection

What we learned. In the assignment, we learned three classification methods, which are both not taught in this course. The basic idea behind decision tree method is really simple and easy to understand. By using bagging and boosting techniques based on decision tree method, we got random forest method and xgboost method. We feel good about learning these methods, and noticed that ensemble learning is a very powerful technique in machine learning.

Shortcomings and what we feel bad. For simplicity, we chose to adjust only one parame-

ter which is max depth for all three methods. We could get better performance or shorter training time if we adjust other parameters. We feel bad about our poor hardware, especially for xgboost method. Although xgboost method provides many parallelizing methods, we could not see the power of it. We tried to use the GPU method for xgboost, but the performance is very similar to non-GPU method on our GTX 1050 ti GPU, i7-7700HQ CPU, and Windows 10 operating system. Our computer even crashed when running on GPU method.

5 Conclusions and future work

5.1 Conclusions

In this experiment, we compared decision tree, random forest and XGBoost methods in fashion-mnist dataset. Decision tree could be seen as the base of Random Forest and XGBoost, and we just got around 81% of accuracy on it. We got very good accuracy score by using Random Forest and XGBoost methods. The highest accuracy we got among the three methods is 89.09% by using XGBoost. However, XGBoost is really time-consuming comparing with the other two methods because of our hardware. We got 87.68% of accuracy which is still a very good score and used much less training time by using Random Forest.

5.2 Future work

In this experiment, we changed only max depth parameter in all three methods, number of trees in random forest method and fixed the others, we may use different parameters in the future, such as max_features in decision tree and random forest, eta which is learning rate in XGBoost.

We may use other tree-based models for comparisons, such as Adaboost tree and MicroSoft's LightGBM. We may also use other models which are not tree-based. Convolutional neural network and ResNet, which both have really good performance in the image field of machine learning, would be good alternatives.

We may upgrade our hardware to see the power of XGBoost, which supports GPU method and can be used distributively. Another alternative is to use cloud, Amazon Web Services and Google Cloud Platform both provide powerful hardware. We may also try to use these methods on spark which is a distributed computing framework. Decision tree and random forest can be found in pyspark.ml package which is DataFrame-based, and spark.mllib which is RDD-based, xgboost also supports spark.

Reference

[1] Chen T, Guestrin C. *Xgboost: A scalable tree boosting system.* 2016.

[2] Choi, J., Song, E., Lee, S., Eungyeol Song, Sangyoun Lee, Jaesung Choi. (2018). L-tree: A local-area-learning-based tree induction algorithm for image classification. Sensors, 18(1), 306. doi:10.3390/s18010306

[3] Goussies, N. A., Ubalde, S., Gomez Fernandez, F., Mejail, M. E. (2014). Optical character recognition using transfer learning decision forests. Paper presented at the 4309-4313. doi:10.1109/ICIP.2014.7025875

[4] Kussul, E., Baidyk, T. (2004). Improved method of handwritten digit recognition tested on MNIST database. Image and Vision Computing, 22(12), 971-981. doi:10.1016/j.imavis.2004.03.008

Appendix

Hardware and Software

- Processor: Intel(R) Core(TM) i7-7700HQ CPU 280 GHz
- Memory: 8 GB
- Coding tool: jupyter notebook
- Interpreter: Python 3.6.4
- Libraries: numpy:1.15.2, matplotlib:3.0.0, scikit-learn:0.20.0, xgboost:0.80

http://www.numpy.org/

https://matplotlib.org/

http://scikit-learn.org/stable/

https://xgboost.readthedocs.io/en/latest/

Data Set

- MNIST-fashion
- https://github.com/zalandoresearch/fashion-mnist

Code Instruction

Code is written in Python 3.6.
Place data set files in the same directory as the code files, use following commands to run the code in terminal
python3 DT.py
python3 RF.py
python3 xgboost.py

or
jupyter notebook DT.ipynb
jupyter notebook RF.ipynb
jupyter notebook xgboost.ipynb

Contribution of group members.

The code was written in team work as each person in charge of particular parts of code, also as the work of data collection and analysis. After all coding work finished, we wrote the report together through using Overleaf. All in a word, each group member did the same contribution.