Assignment 2

Tutors: Jue Wang

Group members:

Haitian Huang (hhua0614), ID: 480263272

Sihong Huang (shua3254), ID: 450000469

Changjin Li (chli3446), ID: 470329711

Abstract

The aim of this project was to develop and operate three machine learning classifier to recognize and classify the grayscale images of costumes. This technique is used in image recognition process which include a wide range of applications from recognizing the face of user in smart devices to distinguish images through giving correspond labels whereby camera. The methodology of three classifiers are decision tree, random forest and K-Nearest Neighbor. The optimal sets of parameters for these classifiers were operated whereby conducting cross validation on training set. The performance of best model choices of each methodology was assessed in a hold-out sample test in terms of accuracy of classification and time of computation. The accuracy score of decision tree classifier is 78.1% and the total computational time is 9.5s. The accuracy score of random forest classifier is 84.7% and the total computational time is 17.4s. The accuracy score of K-Nearest Neighbor classifier is 86.1% and the total computational time is 19.2s. In conclusion, the best performing algorithm for this issue is K-Nearest Neighbor. Despite it had the longest running time, it can produce the highest accuracy among these three classifiers.

1.Introduction

1.1 Objectives

The aim of this project was to develop and operate three machine learning classifier to recognize and classify the grayscale images of costumes. The primary objective was to find the best model in terms of accuracy and total running time by conducting experiments and optimizing parameters on three classifiers.

1.2 Dataset

The dataset we chose is a dataset of Zalando's article images, which consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with 10 classes labels.

1.3 Motivations

Nowadays, the image recognizing techniques is becoming an important component in our daily life. It is applied in diversity areas because of the swift development of science and artificial intelligence. Going from the image recognition search engine to the face recognition on intelligent mobile phone then to cancer cell recognition by X-ray in medical field, image recognition technique has become more and more important.

1.3 Method Overviews

In this project, decision tree, random forest and K-Nearest Neighbor were built and tested by using Fashion-MNIST dataset. Each methodology has its own advantage such as random forest can avoid overfitting and K-Nearest Neighbor is easy to comprehend and operate with high accuracy. These methods were assessed by their optimizing parameters respectively and compared with each other for overall performance.

2.Previous work

The Fashion-MNIST is a dataset of Zalando's article images which is drop-in replacement for original MNIST dataset so that there are lots of the previous work based on it. In this section, focus on three algorithms (KNN, Decision Tree and Random Forest) in previous work with detail on experiment setup and results.

K-Nearest-Neighbors (KNN)

The KNN as a lazy and simple learning method which is widely using in many dataset classifiers. While KNN is lower efficiency than other data classifiers. Guo (2003) has propose a novel KNN to improve K-Nearest-Neighbor classification. Based on KNN is single-handedly to process data so that KNN has a high cost of classifying. The data reduction can help KNN obtain higher accuracy, while when the training set is so large that it will be generate data redundancy. The proposed KNN model is using data reduction and create a model from training data.

To improve efficiency of KNN, the important way is to find a representational model to represent the training data for classification. The Euclidean distance used as the similarity measure ensure same class label can be put together, also new data point cover representative data point is easier to calculate the distance. Using construction algorithm to initial a model, the average result of data reduction is 90.41%, the processing has higher efficiency and lower dependency with k.

Random Forest

M.Pal(2005) has reported an accuracy of 88.37% by using random forest classifier which is better than accuracy of 87.9% on using support vector machine. The dataset that they used was the Landsat-7 Enhanced Thematic Mapper data. There are 2,700 training samples and 2,037 test samples. The sizes of the digits is 307 pixel x 330-pixel.

In the experiment, the number of pixels used for different classes between training and testing. When using 100 trees, the number of features used is 1, 2, 3, 4, 5 and 6. When using three features, selecting different number of trees to variation in classification accuracy, the accuracy decreases from 88.37% to 88.02% when increasing the number of trees. As a result, when the input feature is three and trees around 100, the random forest classifier achieve the highest classification accuracy which is better to that achieved by SVMs.

Decision Tree

Polat(2007) has reported using decision tree to detect epileptic seizure in EEG signals. Based on decision tree classifier, the classification has obtained accuracy of 98.68% and 98.72%. The dataset were collected from healthy people and people who has epilepsy disease. There are five dataset which contains total 500 single channel EEG segments and 4096 test dataset.

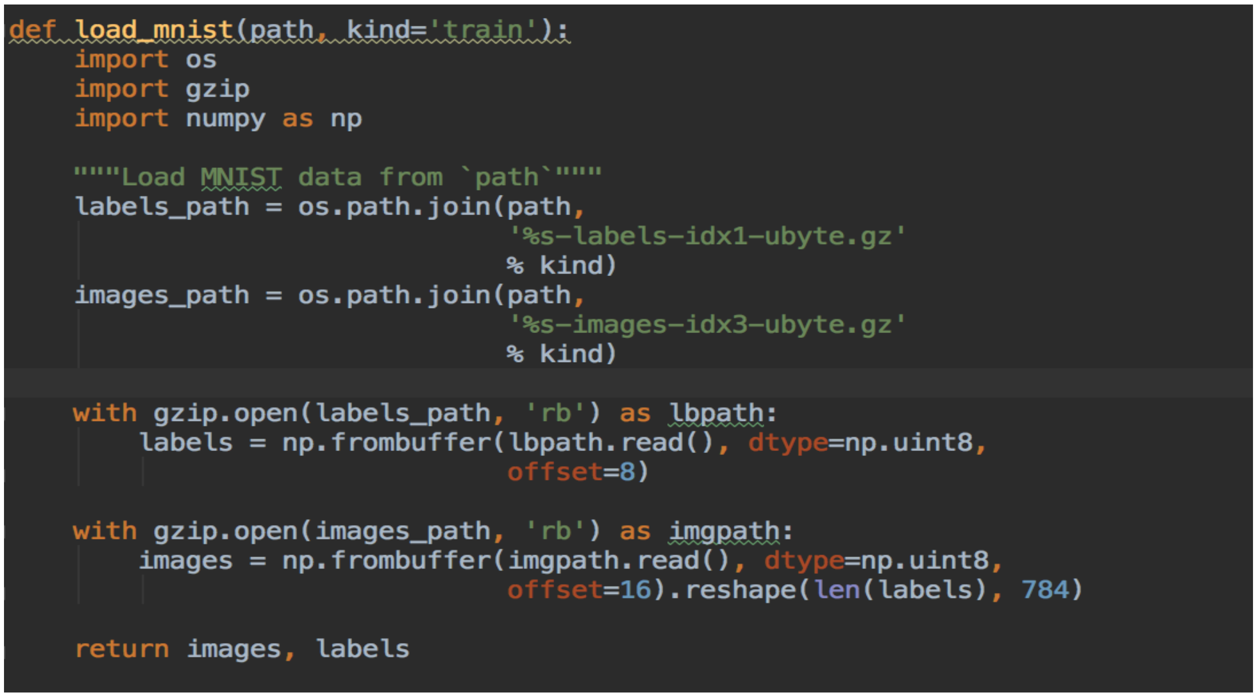
In the experiment, using 5 and. 10-fold cross-validation in the decision tree classifier to training and testing, the data were using 12 bit resolution for 173.61 samples per second, and setting the bang-pass filter as 0.53-40Hz. In the result of cross-validation, the test obtains the highest accuracy of 98.72% which is a efficient tool for accurate decisions.

3.Method

In this section, we will introduce and analyse all the technology which are used in the project such as Pre-processing, Classification and Cross Validation technology. In addition, K-Nearest Neighbor(KNN), Decision Tree and Random Forest are selected as the three classifiers.

3.1 Pre-processing

In generally, before executing the classification algorithm, inputting and handling the selected dataset is the first step. In this project, we choose the MNIST-fashion dataset which includes four files in terms of t10k-images-idx3-ubyte.gz, t10k-labels-idx1-ubyte.gz, train-labels-idx1-ubyte.gz and train-images-idx3-ubyte.gz. Actually we should use the load-mnist(Figure 1) to read these files which have the filename like .gz.



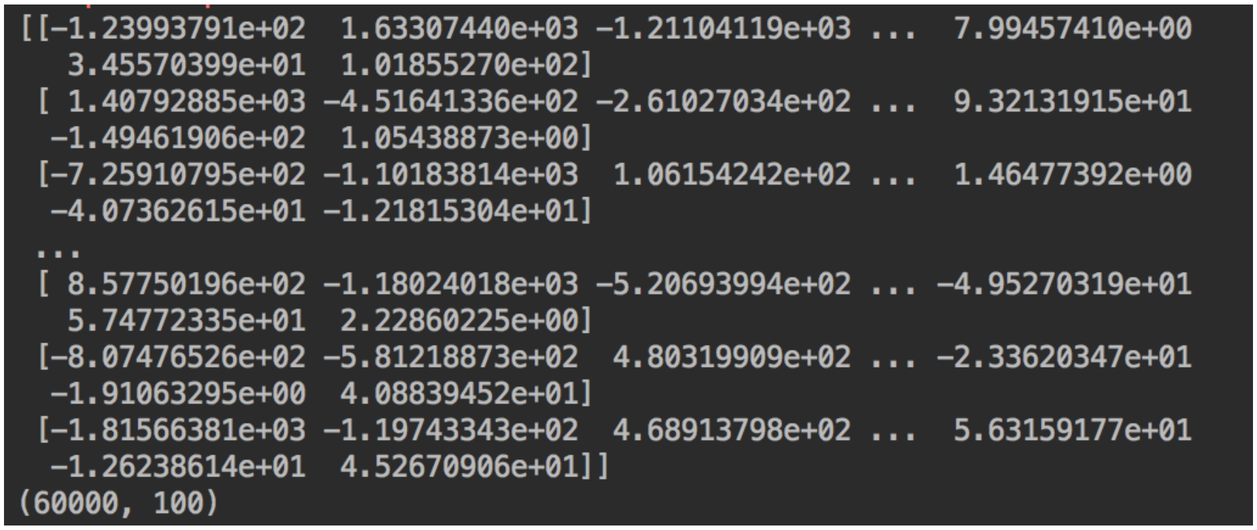
```
def load_mnist(path, kind='train'):
    import os
    import gzip
    import numpy as np

    """Load MNIST data from `path`"""
    labels_path = os.path.join(path,
                               '%s-labels-idx1-ubyte.gz'
                               % kind)
    images_path = os.path.join(path,
                               '%s-images-idx3-ubyte.gz'
                               % kind)

    with gzip.open(labels_path, 'rb') as lbpath:
        labels = np.frombuffer(lbpath.read(), dtype=np.uint8,
                               offset=8)

    with gzip.open(images_path, 'rb') as imgpath:
        images = np.frombuffer(imgpath.read(), dtype=np.uint8,
                               offset=16).reshape(len(labels), 784)

    return images, labels
```

Figure 1: load-mnist

Then, we need to handle the original data which include training data(50000,784), training labels(50000,), testing data(10000,784) and testing labels(10000,) to improve the performance. Principal Components Analysis(PCA) is a great method to handle this data.

Principal Components Analysis(PCA)

In generally, Principal Components Analysis(PCA) and singular value decomposition(SVD) are used to reduce the dimension of data to reduce running time and improve the performance(Wall,2003). The Principal Components Analysis (PCA) can transform the original sample data (M dimension) into the new data (N dimensions). (Figure2)



```
[[-1.23993791e+02  1.63307440e+03 -1.21104119e+03 ...  7.99457410e+00
   3.45570399e+01  1.01855270e+02]
 [ 1.40792885e+03 -4.51641336e+02 -2.61027034e+02 ...  9.32131915e+01
  -1.49461906e+02  1.05438873e+00]
 [-7.25910795e+02 -1.10183814e+03  1.06154242e+02 ...  1.46477392e+00
  -4.07362615e+01 -1.21815304e+01]
 ...
 [ 8.57750196e+02 -1.18024018e+03 -5.20693994e+02 ... -4.95270319e+01
   5.74772335e+01  2.22860225e+00]
 [-8.07476526e+02 -5.81218873e+02  4.80319909e+02 ... -2.33620347e+01
  -1.91063295e+00  4.08839452e+01]
 [-1.81566381e+03 -1.19743343e+02  4.68913798e+02 ...  5.63159177e+01
  -1.26238614e+01  4.52670906e+01]]
(60000, 100)
```

Figure 2: transforming data to 100 dimension

Reducing dimension could be implemented in two methods in terms of the eigenvalue decomposition and the singular value decomposition(SVD). In this report, we focus on analysing the eigenvalue decomposition which reduces the dimension of data set through maintaining the main feature of the original data. Maintaining the feature is achieved by preserving the top N eigenvectors which corresponding the largest N eigenvalue. Preserving the low order principal components and ignoring higher order principal components can reduce the dimension because the low order components often retain the most important aspects of data.

In this project we will use the internal library to implement PCA in terms of *sklearn.decomposition*. The main parameter is ‘n_components=dimension’ which represents the target dimension. In this report, we would not analyze the influence of changing PCA’s parameter and default dimension equal to 100 in below parts.

3.2 Classification design

3.2.1 K-Nearest Neighbor

KNN classifier is operated by measuring the distance between different eigenvalues for classification. K-Nearest Neighbor is simplicity and efficiency(Tan,2006). If most of the K most similar (the nearest neighbor in the feature space) samples of a sample belong to a certain class, then the sample also belongs to this class. K is usually an integer less than 20.In the KNN algorithm, the selected neighbors are all correctly categorization objects. This method determines the class of the unclassified example only in terms of the class of one or more nearest samples in the strategy of class decision.

Algorithmic Process:

For the purpose of achieving KNN, there are primary two categories to process the algorithm. We can operate KNN through Brute-force realization.

Brute-force realization:

To realize the KNN, we can easily associate with it directly. To find k nearest neighbors to do the prediction, then we only need to calculate the distance between the prediction sample and all the samples in the training set, and then calculate the minimum K distances. It is easy to do and conveniently achieve it. But It is effective when the sample size is small and the sample features are few. Therefore, we need to consider a more effective method to realize KNN when the size and features of sample is huge. That is KD-tree we introduce next.

KD-tree realization:

KD tree is a tree with K characteristic dimension. There are 3 steps to achieve KD tree algorithm. The first step is to build up a tree, the second is to search for nearest neighbors, and the last step is to predict. Here is the detailed steps:

- Calculating the variances of N features are calculated from the n-dimensional features of M samples, and the k-dimensional feature nk with the largest variance is used as the root node.
- Generating KD tree to predict sample target points in test sets
- Predicting the category of the most number of classes in the K nearest neighbor.

Parameter Selection:

There are three important elements, the distance measure, the size of k, and the classification rules, which are the three elements of the kNN model.

- Distance measurement.

  There are many ways to measure distance. It is necessary to choose the appropriate distance measure according to the specific circumstances. Commonly used is the Minkowski distance, defined as:

$$D(x, y) = (\sum_{i=1}^{m} |x_i - y_i|^p)^{\frac{1}{p}} \tag{3.1}$$

  In the above formulation, $p \geq 1$

  When p=2 is Euclidean distance, when p=1, it is Manhattan distance.

- The selection of K.

  If the K value is small, it is equivalent to using a training example in a small neighborhood for prediction. In extreme case, k = 1, the test case is only related to the nearest sample, and the training error is very small (0), but if the sample is just noise, the prediction will be wrong, and the test error is very large. That is to say, when the K value is small, the phenomenon of over fitting will occur. If the value of K is large, it is equivalent to using training instances in a large neighborhood to predict, the extreme case is k = n, the result of the test instance is the class with the most instances in the training data set, thus resulting in under-fitting. For this dataset, we will use cross-validation to select the appropriate K.

- The classification decision rule.

  The classification decision rule in kNN is usually a majority vote, that is, the class of test samples is determined by the majority of k neighboring samples of test samples. The majority voting rule is explained as follows: given the test sample x, its nearest K training instances constitute the set Nk(x), and the classification loss function is 0-1 loss. If the category of Nk(x) covers cj, then the classification error rate is:

$$\frac{1}{k} \sum_{x_i \in N_k(x)} I\{y_i \neq c_j\} = 1 - \frac{1}{k} \sum_{x_i \in N_k(x)} I\{y_i = c_j\} \tag{3.2}$$

  In order to minimize the classification error rate(the empirical risk), so most votes are equivalent to the empirical risk minimization. The model of kNN is equivalent to getting Nk(x) for any x, the loss function is

0-1 loss, and the optimization strategy is the least empirical risk. In general, a majority vote is applied to the samples in Nk(x).

3.2.2 Decision Tree

Decision Tree is a non-parametric supervised learning method for classification and regression. The goal is to create a model that predicts the value of the target variable by deriving simple decision rules from the data properties.

Decision tree is a tree structure. Generally, a decision tree contains one root node (Initial sample size), several internal nodes (Each internal node corresponds to one attribute test, that is, one decision.) and several leaf nodes (The extreme end of one direction of a tree, representing the output of the result.).

From the root node to each leaf node, each decision sequence is formed; our purpose of training decision tree classifier is to produce a decision tree with strong generalization ability, that is, the decision tree has a strong ability to deal with unknown example. Its basic process follows the strategy of separating and processing respectively:

Algorithmic Process:

- Enter sample set D {(x1, y1), (x2, y2),..., (xn, yn)}, attribute set A {a1, a2,..., ad}, and all sample sets are stored in the root node
- A best attribute A1 is selected from attribute set A by certain rules (specific rules are determined by algorithm such as ID3, CART). All samples flow from the root node to the decision node. According to the value of the sample on the attribute a1, the flow direction corresponds to the corresponding direction.
- After all the attributes are utilized whereby the above step, a complete tree is formed, which passes through all the attributes on each judgment path. At this stage, the output class of all leaf nodes is defined as the one with the largest proportion (using the prior distribution) of the samples arriving at the leaf node in the training process. At this point, a tree decision training is completed.

Variable Importance:

The key of decision tree learning is how to choose the optimal partitioning attributes. It is considered that as the partitioning process proceeds, the samples contained in the branch nodes of the decision tree belong to the same class as possible, which means the purity of the nodes becomes higher and higher. There are several different rules for measuring the purity of samples, which also produce different decision tree algorithms respectively.

Information Gain (ID3 Algorithm):

The ID3 algorithm uses information gain as an impurity. Firstly, the concept of information entropy is necessary to be introduced. Information entropy is the most commonly used index for measuring the purity of a sample set. Assuming that the proportion of class k samples in the current sample set *D* is p_k(k=1,2,...,|y|), the information entropy of *D* is defined as:

$$Ent(D) = -\sum_{k=1}^{|y|} p_k \log_2 p_k \tag{3.3}$$

The smaller the *Ent(D)* is, the higher the purity of *D* is. And |*y*| denotes the number of possible values of attributes. Assuming that there are *V* possible values $\{a^1,a^2,...,a^V\}$ for the discrete attribute *a*, *a* is used to partition the sample set *D* to produce *V* branch nodes, where the Vth branch node flows into the sample where all attributes a in *D* have a value of a^V, which is recorded as D^V, then attribute *a* partitions *D*. The information gain obtained by dividing is

$$Gain(D,a) = Ent(D) - \sum_{v=1}^{V} |D^V|/|D| \cdot Ent(D^V) \tag{3.4}$$

Ent(D) is the information entropy Where $|D^V|$ refers to the number of samples taken from *D* attribute in *a* attribute, $|D^V|$ / $|D|$ can be regarded as the weight in a^V direction. The greater the information gain, the greater the "purity enhancement" of the partition using a attribute. For example. the current optimal partition:

$$a^* = arg \max_{a \in A} Gain(D,a)$$

(3.5)

Gain Rate (C4.5 Algorithm):

The C4.5 algorithm uses gain rate as an impurity. C4.5 is an extension and optimization of ID3 algorithm. It overcomes the disadvantage of ID3 algorithm that the information gain tends to select attributes with multiple attribute values as splitting attributes. It can handle discrete and continuous attribute types and discretize the continuous attributes. The gain rate is defined as:

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

(3.6)

And *IV(a)* is defined as:

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

(3.7)

IV(a) is called the intrinsic value of attribute a, and the larger the number of possible values of attribute a (the greater the V), the larger the value of *IV(a)* will usually be; compared with information gain, gain rate has a preference for attributes with fewer values of attribute. Therefore, C4.5 algorithm does not directly use the gain rate of all attributes as a basis for comparison, it has a heuristic procedure: First selecting the attributes whose information gain is higher than the average level in candidate partitioning attributes, and then select the attributes with the highest gain rate from them.

Gini Coefficient (CART Algorithm):

The CART algorithm uses gini coefficient as an impurity. The purity of data D can be measured by Gini value as below:

$$Gini(D) = \sum_{k=1}^{|y|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|y|} p_k^2$$

(3.8)

Gini(D) reflects the probability that two samples are extracted from a data set D. The smaller the *Gini(D)*, the higher the purity of the data set D. For a property a, the Gini index is shown as below:

$$Gini_{index(D,a)} = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v)$$

(3.9)

So in the set A of candidate attributes, the current optimal partitioning attributes are selected which have the smallest Gini index among the current remaining attributes, That is:

$$a^* = arg \max_{a \in A} Gain\_index(D,a)$$

(3.10)

Parameter Selection:

The classification accuracy of decision tree is significantly depended on:

- Minimum number of leaf nodes(In scikit-learn the parameter is min_samples_leaf).
- Maximum depth of decision tree(In scikit-learn the parameter is max_depth).

These two parameters prevent over-fitting. For the minimum number of leaf nodes. If the number of leaf nodes is smaller than the number of samples, it will be pruned with the sibling nodes. And for the maximum depth of decision tree, this parameter controls the size of the tree.If the depth is too deep, it will be difficult for us to understand.

Classifier Optimization:

In order to classify training samples as accurately as possible, the process of node partitioning is iterated repeatedly, which sometimes produces too many branches of the decision tree. At this time, some characteristics of the training set may be regarded as general properties of all data because of over-learning of training set, which possibly lead over-fitting. Therefore, we need to consider pruning to actively remove some branches to reduce the risk of over-fitting.

Pre-pruning

In the process of decision tree generation, the performance of each node is estimated before partitioning. If the partitioning of the current node cannot improve the generalization performance of the decision tree, the partitioning is stopped and the current node is marked as a leaf node.

Here is the process steps:

- in order to measure generalization ability, we use the retention method to divide the sample set into training set and validation set.
- According to the information gain criterion, a^* is selected as the first non-leaf node under the root node. Training the model classified by this attribute and the model which use this node as leaf node respectively. Comparing the accuracy of the two models on the verification set, and the better scheme is selected.
- Repeat the above step to inspect all attributes until the final decision tree is completed.

Advantages: Pre-pruning makes many branches of the decision tree unwrapped, reduces the risk of over-fitting, and significantly reduces the training time and testing time of the decision tree(Quinlan,1987).

Disadvantages: While current partitioning of some branches does not enhance generalization capabilities, it may even cause temporary degradation of generalization capabilities, subsequent partitioning on the basis of which may lead to significant performance improvements.

Post-pruning

A complete decision tree is generated from the training set, and then the non-leaf node is examined from bottom to top. If the substitution of the corresponding subtree of the node for the leaf node can improve the generalization ability of the decision tree, the subtree is replaced by the leaf node.

Here is the process steps:

- For a complete decision tree using all attributes, which is formed without any pruning and only based on a certain information purity evaluation method, starting from the last non-leaf node, training the model with pruning this node and the model without pruning this node respectively, then comparing the generalization ability of them.
- if the generalization ability is improved, the corresponding model change or maintenance operation will be adopted.

- repeat the above procedure until all the non-leaf nodes complete the pruning effect evaluation.

Advantages: Under-fitting risk is very small, generalization ability of it is often better than pre-pruning decision tree.

Disadvantages: The post-pruning process is performed after the complete decision tree is generated, and all the non-leaf nodes in the tree are examined one by one from the bottom up, so the training time is expensive.

Pros & Cons:

Advantages:

- Simple to understand and to interpret. Trees can be visualised.
- Needing little data preparation is required. Other technologies usually require data normalization, creating virtual variables and removing null values.
- Decision tree is able to process digital and categorical data. Other classification are usually used to analyze datasets with only one variable type.
- Able to handle multi-output problems
- Statistical tests can be used to verify the model, which is to verify the reliability of the model.

Disadvantage:

- Decision tree algorithm can create complex trees, but there is no basis for generalization, which is over-fitting. To avoid this problem, we can execute pruning, that is setting the minimum number of leaf nodes required or the maximum depth of the tree.
- There are some difficult concepts to learn because decision trees can't easily express them, such as XOR, parity check, or multiplexer problems

And the another important optimization for decision tree is random forest. Next we will introduce the detailed random forest classifier.

3.2.3 Random forest

In machine learning, random forest is an ensemble learning method for classification that operates by construction a multitude of decision trees at training time and output its class by model of classes. The random forest has the following characteristics:

- It is unexcelled in accuracy among current algorithms
- It runs efficiently on large dataset.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.

Algorithm

Tree Bagging

The algorithm for random forests applies the general technique of bootstrap aggregating. Given a training set $X = x_1,\ldots,x_n$ with responses $Y = y_1,\ldots,y_n$. Bagging mean a random sample is taken with replacement of the training set and fits tress to these samples:

For $b = 1,\dots,B$:

1. Sample with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b.

After B times training, predictions for unseen sample X can be made by averaging the predictions from all the individual regression tress X.

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x') \tag{3.11}$$

This bootstrapping procedure can be improving model performance due to decreasing the variance of the model and without increasing the bias. So single tree is highly sensitive to noise in the predictions of training set. The average of many trees is not due to the trees are not correlated. Training many trees on a single training set can have a strongly correlated tree. Also, all the individual regression trees can evaluate the uncertainty of the prediction as the standard deviation on X.

The above procedure is the original bagging algorithm in trees. Random forests have a specially

Way is the improve the tree learning algorithm. The most important advantages of random forest that does not need to cross validation or using single training set to obtain unbiased estimation.

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}. \tag{3.12}$$

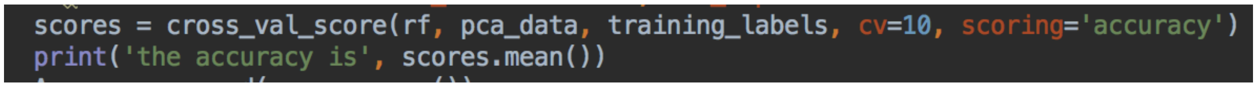
The processing of random forest is similar with bagging. First, using bootstraping random choose m training data from original dataset and performs n-tree samples to generate n-tree training sets. Based on n-tree training sets, train n-tree decision tree models separately. For a single decision tree model, assuming that the number of training sample feature is N, then process the splitting according to information gain ratio. Every tree has been split until all the training sets belong to same class. Also, it is unnecessary to pruning in the decision tree splitting. To generated multiple decision trees into random forest. For classification of random forest, the result is determined by votes of the multi-tree classifier. For the regression problem, the result is determined by predicted average values of the multiple trees.

3.3 Cross Validation

Cross-validation is a technique to evaluate models by partitioning the original training data into a training set and a validation set(Kohavi,1995). Using the training set train the model. Then using the validation set evaluate it.

In k-fold cross-validation, the training data is randomly partitioned into k equal size data. Firstly, choosing the random k-1 subsamples are used for training data. Then, the remaining single subsample is retained as the validation set for evaluating the mode. After that, in order to proving each of sub-dataset are exactly used as the validation, repeating k times execute the cross-validation process. All the data is used for validation is one of the advantage of this method.

In this report, we will use the 10-fold cross-validation to evaluate the model. We would import *sklearn.model_selection.* The below figure show the executing code in terms of 10-fold cross-validation. The scores in Figure 3 include 10 values of accuracy. We should get the mean of these values.



```
scores = cross_val_score(rf, pca_data, training_labels, cv=10, scoring='accuracy')
print('the accuracy is', scores.mean())
```

Figure 3: Executing 10-fold cross-validation

4.Experiments and Discussion

4.1 Experiments

All the experiment details of these three classifications would be described and analyzed in this section. This project would run in ***python 3.6*** on ***Macbook Pro*** with 2.3 GHz ***Intel Core i5*** CPU.

K-Nearest Neighbor(KNN): imported from ***sklearn.neighbors***.

Decision Tree: imported from ***sklearn.tree***.

Random Forest: imported from ***sklearn.ensemble***.

4.1.1 Analysing method (Variable controlling)

In order to explore the influence of one parameter changing, controlling several other parameters are important. Thus, the method of variable controlling can help us analyze the relationship between parameters and results in these classifications.

4.1.2 K-Nearest Neighbor

(1)Find the best model(the best k) using 10-fold Cross Validation(assuming reduce dimension to 100)

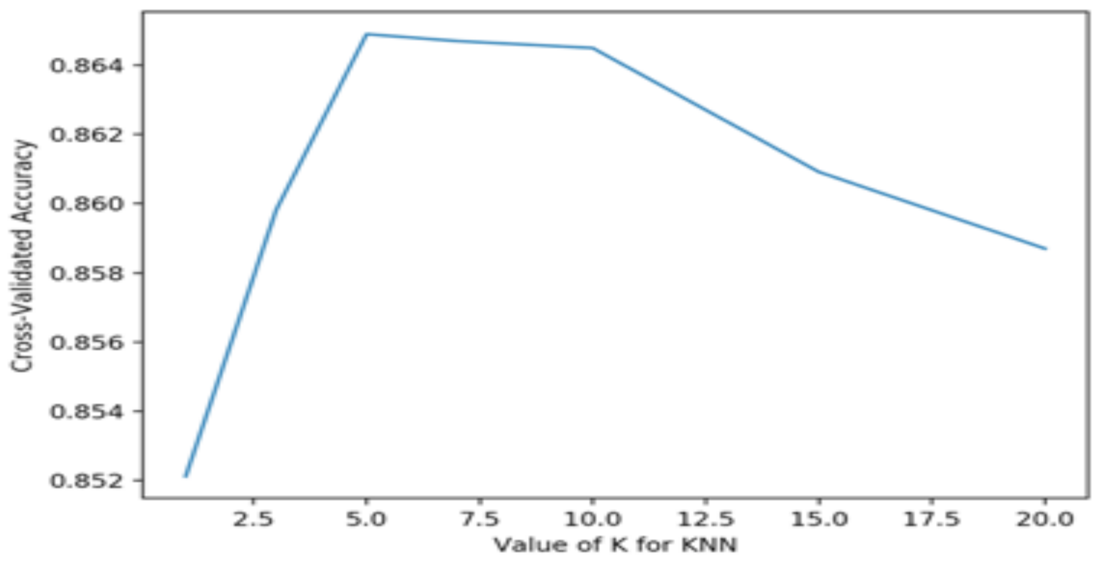


Figure 4: Cross-Validated in KNN

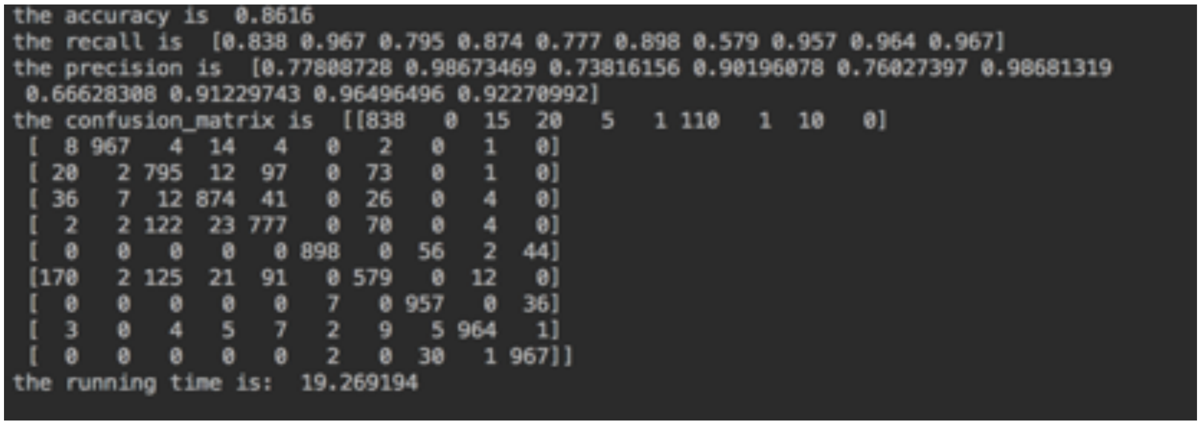
Table 1: Cross-Validated in KNN

k	1	3	5	7	10	15	20
Mean Accuracy	85.2%	85.9%	86.4%	86.4%	86.4%	86.0%	85.8%
Time	77.2	93.6	102.1	106.9	112.7	120.0	122.4

According to the 10-fold Cross Validation, we can get the above data. It is obvious when k equal to 5, the accuracy reach the peak. There is a guess when the k more than a special value, the accuracy will decrease because of some interference data being selected. In addition, with the increasing of k, there is a increasing trend in terms of running time. Overall, we guess that k =5 is the best training model.

(2) Using the best model k= 5 predict testing data.

The below figure show the accuracy, the recall, the precision and the confusion matrix. In KNN algorithm, predicting the label 1 and label 9 has the highest accuracy which the recall reach 0.967. In addition, the accuracy of predicting label 8 is also great which get 0.964. However, the label 6 has the lowest accuracy at 0.666.



```
the accuracy is  0.8616
the recall is  [0.838 0.967 0.795 0.874 0.777 0.898 0.579 0.957 0.964 0.967]
the precision is  [0.77808728 0.98673469 0.73816156 0.90196078 0.76027397 0.98681319
 0.66628308 0.91229743 0.96496496 0.92270992]
the confusion_matrix is  [[838   0  15  20   5   1 110   1  10   0]
 [  8 967   4  14   4   0   2   0   1   0]
 [ 20   2 795  12  97   0  73   0   1   0]
 [ 36   7  12 874  41   0  26   0   4   0]
 [  2   2 122  23 777   0  70   0   4   0]
 [  0   0   0   0   0 898   0  56   2  44]
 [170   2 125  21  91   0 579   0  12   0]
 [  0   0   0   0   0   7   0 957   0  36]
 [  3   0   4   5   7   2   9   5 964   1]
 [  0   0   0   0   0   2   0  30   1 967]]
the running time is:  19.269194
```

Figure 5: Result of predicting using KNN (k=5)

4.1.3 Decision Tree

(1) find the best model using 10-fold Cross Validation(assuming reduce dimension to 100)

- Find the best depth

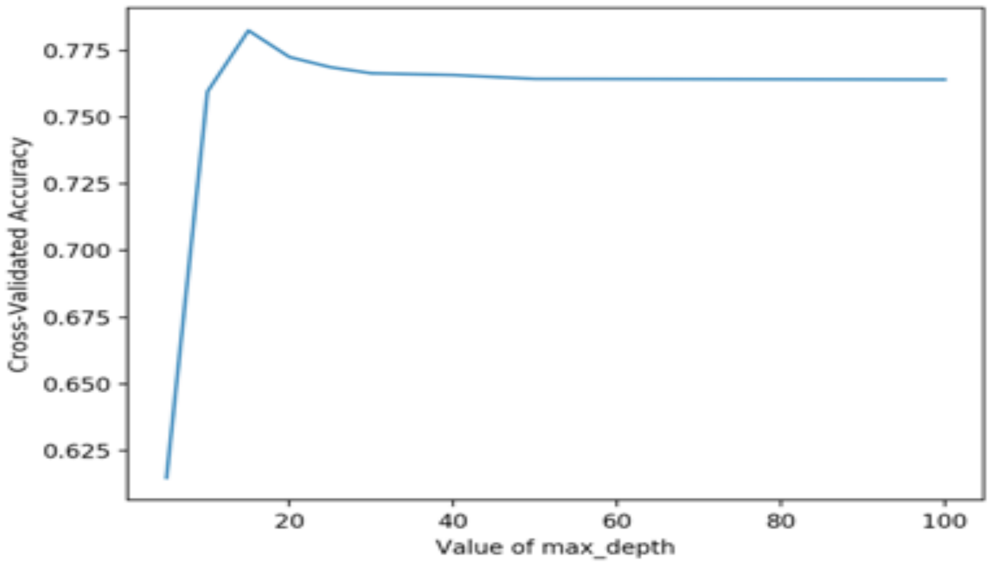


Figure 6: Cross-Validated in Decision Tree

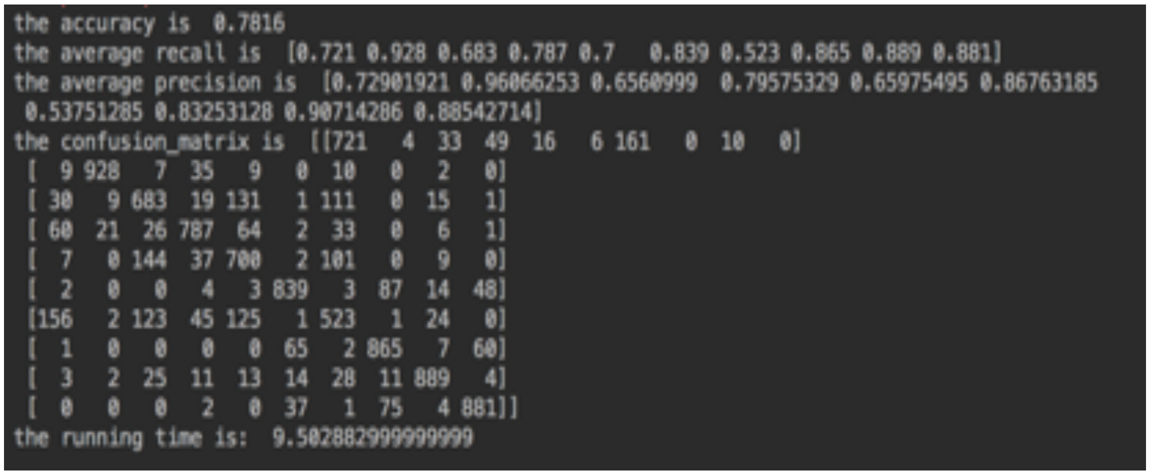
Table 2: Cross-Validated in Decision Tree

Depth	5	10	15	20	25	30	40	50	100
Mean Accuracy	61.4%	75.9%	78.2%	77.2%	76.8%	76.6%	76.5%	76.4%	76.4%
Time	33.6	62.8	85.5	103.6	119.8	125.5	126.6	133.7	134.9

In this experiments, we chose the depth_range [5,10,15,20,25,30,40,50,100]. When the depth equal to 5, the accuracy is only 61.4%. We guess that the depth too small may occur the underfitting. In addition, it is easy to see when the depth more than a special value, the accuracy will decrease. In the Decision Tree algorithm, if the depth is too large, the phenomenon of overfitting may occur. According to the above data, when depth equal to 15, the accuracy reach the peak and the running time is acceptable.

(2) Using the best model depth=15 predict testing data.

The below figure show the accuracy, the recall, the precision and the confusion_matrix when depth equal to 15. The accuracy of using DT model are not too high at 0.781. The data about label 1 is greatest to recognise. However, the accuracy of label 2 and label 6 should be improved.



```
the accuracy is  0.7816
the average recall is  [0.721 0.928 0.683 0.787 0.7   0.839 0.523 0.865 0.889 0.881]
the average precision is  [0.72901921 0.96066253 0.6560999  0.79575329 0.65975495 0.86763185
 0.53751285 0.83253128 0.90714286 0.88542714]
the confusion_matrix is  [[721   4  33  49  16   6 161   0  10   0]
 [  9 928   7  35   9   0  10   0   2   0]
 [ 30   9 683  19 131   1 111   0  15   1]
 [ 60  21  26 787  64   2  33   0   6   1]
 [  7   0 144  37 700   2 101   0   9   0]
 [  2   0   0   4   3 839   3  87  14  48]
 [156   2 123  45 125   1 523   1  24   0]
 [  1   0   0   0   0  65   2 865   7  60]
 [  3   2  25  11  13  14  28  11 889   4]
 [  0   0   0   2   0  37   1  75   4 881]]
the running time is:  9.502882999999999
```

Figure 7: Result of predicting using DT (depth=15)

4.1.4 Random Forest

(1) find the best model using 10-fold Cross Validation(assuming reduce dimension to 100)

- Find the best value of n_estimators(using the Variable controlling assuming the max_depth=25)

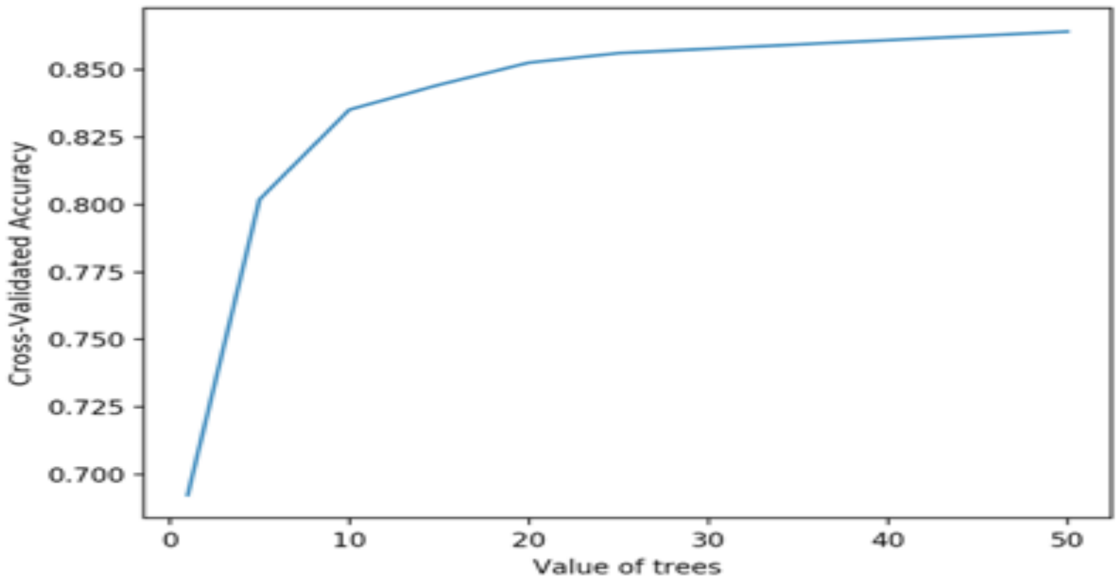


Figure 8: Cross-Validated in Random Forest(n_estimators)

Table 3: Cross-Validated in Random Forest(n_estimators)

N_estimators	1	5	10	15	20	25	50
Accuracy	69.2%	80.1%	83.5%	84.4%	85.2%	85.6%	86.4%
Time	6.9	32.0	63.0	89.7	122.7	153.2	308.1

Using 10-fold Cross Validation, we choose the n_estimators=[1,5,10,15,20,25,50]. The parameter of n_estimators represent the number of trees in the Random Forest algorithm. As the above data, when n_estimators equal to 25, the accuracy is great and the running time is acceptable. Moreover, with the increasing of n_estimators, there is a sharply increase in terms of running time. Thus, we chose the n_estimators = 25 in the next step.

- Find the best max_depth(according to the last part, we chose the n_estimators = 25)

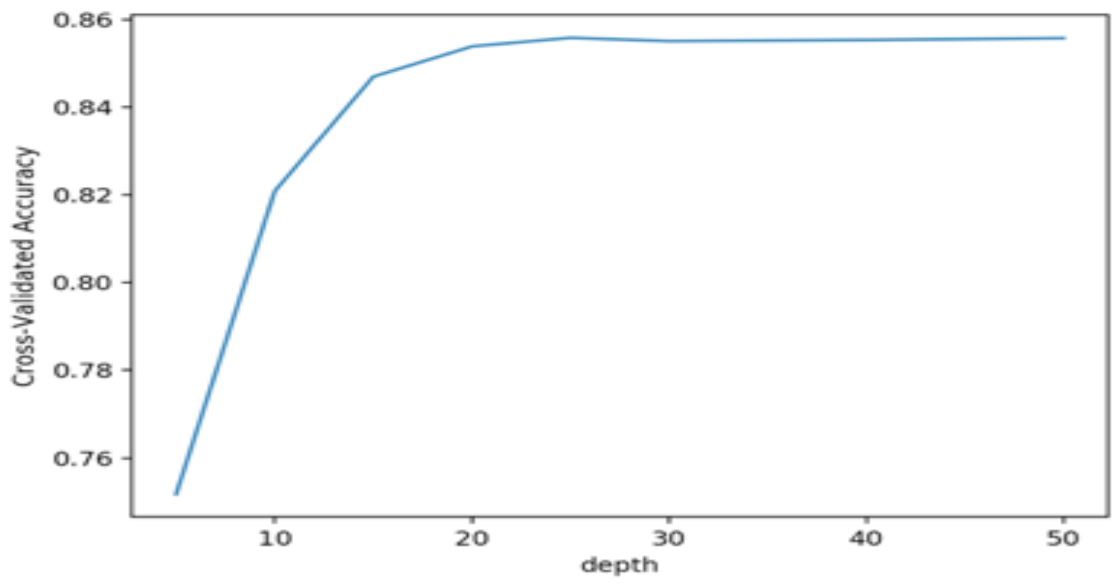


Figure 9: Cross-Validated in Random Forest(Max_depth)

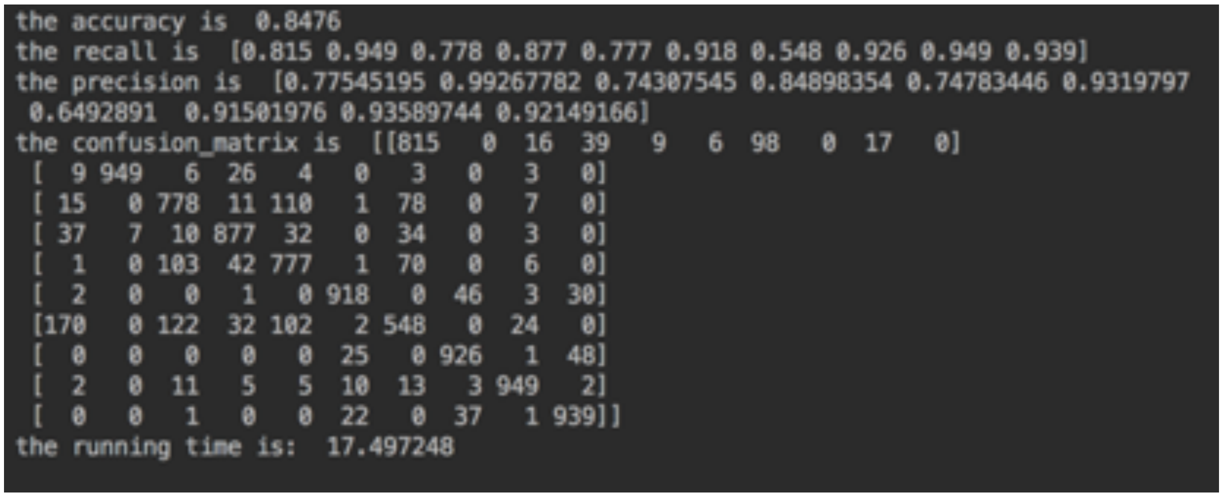
Table 4: Cross-Validated in Random Forest(Max_depth)

Max Depth	5	10	15	20	25	30	40	50
Accuracy	75.1%	82.0%	84.6%	85.3%	85.5%	85.5%	85.5%	85.5%
Time	57.3	99.3	131.4	151.4	156.7	168.1	162.2	163.5

We choose the Max_depth=[5,10,15,20,25,30,40,50] to find the best model using 10-fold Cross Validation. As the above data showed, the accuracy reach the peak at 85.5% when Max_depth equal to 25. According to Random Forest algorithm, if the value of Max_depth is too larger, the overfitting phenomenon may occur. In this data, we find that when the Max_depth larger than 25, the accuracy is trend to stable.

(2) Using the best model n_estimators=25, max_depth=25 predict testing data.

The below figure show the accuracy, the recall, the precision and the confusion_matrix when max_depth and n_estimators equal to 25. The accuracy is great reach 0.847 when using Random Forest Algorithm. The label 1 and label 8 data is easy to recognise which accuracy reach 0.949. In addition, the label 6 data is hardest to recognise.



```
the accuracy is  0.8476
the recall is  [0.815 0.949 0.778 0.877 0.777 0.918 0.548 0.926 0.949 0.939]
the precision is  [0.77545195 0.99267782 0.74307545 0.84898354 0.74783446 0.9319797
 0.6492891  0.91501976 0.93589744 0.92149166]
the confusion_matrix is  [[815   0  16  39   9   6  98   0  17   0]
 [  9 949   6  26   4   0   3   0   3   0]
 [ 15   0 778  11 110   1  78   0   7   0]
 [ 37   7  10 877  32   0  34   0   3   0]
 [  1   0 103  42 777   1  70   0   6   0]
 [  2   0   0   1   0 918   0  46   3  30]
 [170   0 122  32 102   2 548   0  24   0]
 [  0   0   0   0   0  25   0 926   1  48]
 [  2   0  11   5   5  10  13   3 949   2]
 [  0   0   1   0   0  22   0  37   1 939]]
the running time is:  17.497248
```

Figure 10: Result of predicting using RF (n_estimators=25 ,Max_depth=25)

4.2 Discussion and personal reflection

For three different classifiers in this report, we made up the below matrix to show the performance. (This running time is not including the time of handling dataset)

Table 5: Performance comparison of three classifiers

Type of Classifiers	accuracy	Running time
KNN	86.1%	19.2s
Decision Tree	78.1%	9.5s
Random Forest	84.7%	17.4s

It is obvious that using KNN model have the best accuracy in predicting Fashion_MNIST dataset. Meanwhile, the lowest accuracy is 78.1% when using Decision Tree Algorithm. As we mentioned in part 3, the performance of Decision Tree may not so great when the dataset has multiple classes. In this dataset, we can try to use more trees to fit training data. Random Forest Algorithm is the optimization of Decision Tree which the accuracy can reach 84.7% in predicting Fashion_MNIST dataset. In addition, comparing to KNN, Random Forest and Decision Tree have less calculated amount. Thus, Random Forest and Decision Tree have faster running time in terms of 17.4s and 9.5s.

5.Conclusion and future work

Overall, we guess Random Forest is the greatest algorithm in these three algorithm to predict Fashion_Mnist dataset. In this experiment, although using KNN has the best accuracy, it has large calculated amount which would cause longer running time. In addition, the accuracy of using Decision Tree is too low, compared with using Random Forest. We guess the multiple classes problems could not be solved perfectly by using Decision Tree. For the future work, we will research how to implement the optimization to solve overfitting problems in Decision Tree and Random Forest. In addition, we will try other parameters in these algorithms and other algorithm like SVM to predict this dataset.

Contribution

Haitian Huang: Writing the correlated part of decision tree classifier. Writing the abstract and introduction. Adjusting the layout and font of whole project.

Changjin Li: Writing the correlated part of K nearest neighbor classifier. Writing the conclusion and personal reflection. Drawing the graphs and figures. Controlling the parameters to find and test the best model.

Sihong Huang: Writing the correlated part of random forest classifier. Writing and concluding the correlated part of reference.

Reference

[1] Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217-222.

[2] Polat, K., & Güneş, S. (2007). Classification of epileptiform EEG using a hybrid system based on decision tree classifier and fast Fourier transform. *Applied Mathematics and Computation*, *187*(2), 1017-1026.

[3] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003, November). KNN model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"* (pp. 986-996). Springer, Berlin, Heidelberg.

[4] Ding C, He X. K-means clustering via principal component analysis. InProceedings of the twenty-first international conference on Machine learning 2004 Jul 4 (p. 29). ACM.

[5] Wall, M. E., Rechtsteiner, A., & Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis* (pp. 91-109). Springer, Boston, MA

[6] Tan, S. (2006). An effective refinement strategy for KNN text classifier. *Expert Systems with Applications*, *30*(2), 290-298.

[7] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81-106.

[8] Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*, *43*(6), 1947-1958.

[9] Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*(Vol. 14, No. 2, pp. 1137-1145).

[10] Quinlan, J. R. (1987, August). Generating production rules from decision trees. In *ijcai* (Vol. 87, pp. 304-307).

Appendix

Code Instructions

Files:
decision_tree.py: The code to run decision tree classifier.
k_nearest_neighbor.py: The code to run k_nearest_neighbor classifier.
random_forest.py: The code to run random_forest classifier.

Dataset:

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

Running steps:

1. Import the Fashion-MNIST dataset include 4 files:
- *train-images-idx3-ubyte.gz*
- *train-labels-idx1-ubyte.gz*
- *t10k-images-idx3-ubyte.gz*
- *t10k-labels-idx1-ubyte.gz*
2. Running k_nearest_neighbor.py file which include using the Cross-Validation choose the greater parameter and KNN testing.
3. Running decision_tree.py as step2
4. Running random_forest.py as step2

When running the **decision_tree.py** file, please set the working directory in the "code" folder.
When running the **k_nearest_neighbor.py** file, please set the working directory in the "code" folder.
When running the **random_forest.py** file, please set the working directory in the "code" folder.