CSE 601 Project3 Cluster report

Group Member

Xiaoixn Wu, UB ID: xwu36, Personal #:50208041
Xiaowei Shen, UB ID: xshen5, Personal #:50206168
Sheng-yung Cheng, UB ID: shengyun, Personal #:50207916

Results:

Table 1

	project3_dataset1					project3_dataset2				
	Accuracy	Precision	Recall	F-1	Running time	Accuracy	Precision	Recall	F-1	Running time
KNN	0.9736	0.9955	0.9368	0.9647	31.19s	0.7035	0.6337	0.3607	0.4434	6.640s
DT	0.9331	0.9179	0.8987	0.9067	106.53s	0.6773	0.5510	0.4127	0.4615	9.582s
NB	0.9349	0.9179	0.9044	0.9100	18.3s	0.7035	0.5709	0.6159	0.5867	3.8s
RF	0.9613	0.9896	0.9038	0.9440	31.015s	0.7033	0.6418	0.349	0.4447	10.59s
BDT	0.9789	0.9957	0.9511	0.9722	23.31s	0.6968	0.5715	0.4864	0.5099	7.665s

KNN: K Nearest Neighbor
DT: Decision Tree
NB: Naïve Bayes
RF: Random Forests
BDT: Adaboost Decision Tree:

1. K Nearest Neighbor:

Algorithm:

KNN makes predictions using the training dataset directly.
Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances.
We use heap structure to quickly find the top K nearest neighbors in O(Klog(N))
https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/

Parameter setting:

K	project3_dataset1				
	Accuracy	Precision	Recall	F-1	Running time
3	0.9648	0.9747	0.9270	0.9488	33.00s
5	0.9684	0.9814	0.9341	0.9558	30.59s
7	0.9718	0.9811	0.9433	0.9613	30.95s
9	0.9718	0.9870	0.9383	0.9613	29.80s
11	0.9701	0.9870	0.9363	0.9604	30.58s
13	0.9718	0.9913	0.9363	0.9625	30.32s
15	0.9736	0.9955	0.9368	0.9647	31.19s
17	0.9701	0.9952	0.9273	0.9594	31.15s
19	0.9718	0.9952	0.9320	0.9619	29.77s

K: 15 is chosen as the value of K.
We did lots of tests on K and found 15 achieves the best balance between the efficiency and accuracy. As K increases, we can see the accuracy increases first and when it reaches 15, it starts to go down.

Numerical attribute & Categorical attribute:

For numeric data we use Euclidean distance to determine which of the K instances in the training dataset are most similar to a new input.

Euclidean Distance(x, xi) = sqrt(sum((xj – xij)^2))

And for categorical data, the distance measure equations are shown as follows:

If they are exactly the same, the distance is 0, otherwise the distance is 0

And all the distances are normalized by being divided by the |max - min|

Results:

From table1 we can see:

- KNN has the second best predicted results on dataset1, only worse than Adaboost Decision Tree, and also it performs very good on dataset2.
- The calculating time of KNN is also not very fast. It ranks at the third place out of five.

Cross Validation Result of dataset1:

```
-------------------------------------------round: 0-----------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9565, f-1 score = 0.9778
-------------------------------------------round: 1-----------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9412, f-1 score = 0.9697
-------------------------------------------round: 2-----------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9286, f-1 score = 0.9630
-------------------------------------------round: 3-----------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9524, f-1 score = 0.9756
-------------------------------------------round: 4-----------------------------------------------
accurcay = 0.9649, precision = 1.0000, recall = 0.9000, f-1 score = 0.9474
-------------------------------------------round: 5-----------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9630, f-1 score = 0.9811
-------------------------------------------round: 6-----------------------------------------------
accurcay = 1.0000, precision = 1.0000, recall = 1.0000, f-1 score = 1.0000
-------------------------------------------round: 7-----------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9333, f-1 score = 0.9655
-------------------------------------------round: 8-----------------------------------------------
accurcay = 0.9123, precision = 1.0000, recall = 0.8387, f-1 score = 0.9123
-------------------------------------------round: 9-----------------------------------------------
accurcay = 0.9643, precision = 0.9545, recall = 0.9545, f-1 score = 0.9545

result:
accurcay = 0.9736, precision = 0.9955, recall = 0.9368, f-1 score = 0.9647

--- 31.19899606704712 seconds ---
```

Cross Validation Result of dataset2:

```
-------------------------------------------round: 0-----------------------------------------------
accurcay = 0.6383, precision = 0.6111, recall = 0.5238, f-1 score = 0.5641
-------------------------------------------round: 1-----------------------------------------------
accurcay = 0.7447, precision = 0.5455, recall = 0.4615, f-1 score = 0.5000
-------------------------------------------round: 2-----------------------------------------------
accurcay = 0.7174, precision = 0.7500, recall = 0.4737, f-1 score = 0.5806
-------------------------------------------round: 3-----------------------------------------------
accurcay = 0.7174, precision = 0.5833, recall = 0.4667, f-1 score = 0.5185
-------------------------------------------round: 4-----------------------------------------------
accurcay = 0.5652, precision = 0.5000, recall = 0.1000, f-1 score = 0.1667
-------------------------------------------round: 5-----------------------------------------------
accurcay = 0.6522, precision = 0.6250, recall = 0.2778, f-1 score = 0.3846
```

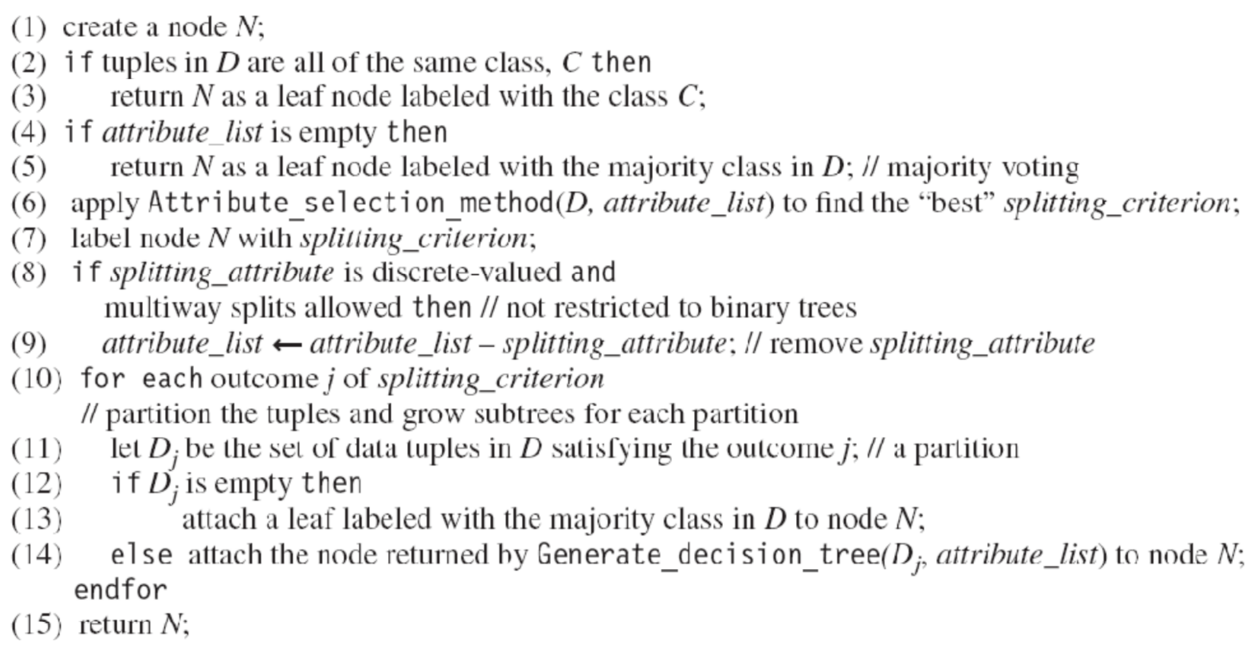
```
------------------------------------------round: 6------------------------------------------------
accurcay = 0.8261, precision = 1.0000, recall = 0.3846, f-1 score = 0.5556
------------------------------------------round: 7------------------------------------------------
accurcay = 0.7826, precision = 0.5556, recall = 0.4545, f-1 score = 0.5000
------------------------------------------round: 8------------------------------------------------
accurcay = 0.6957, precision = 0.6667, recall = 0.2500, f-1 score = 0.3636
------------------------------------------round: 9------------------------------------------------
accurcay = 0.6957, precision = 0.5000, recall = 0.2143, f-1 score = 0.3000

result:
accurcay = 0.7035, precision = 0.6337, recall = 0.3607, f-1 score = 0.4434

--- 6.640501022338867 seconds ---
```

Conclusions:
Pros:
1. Ease to interpret output
2. Calculating time is short
3. Easy to be implemented
4. Robust to noisy training data
Cons:
1. Predictive Power is relatively weak
2. Need to determine value of parameter K
3. Distance based learning is not clear which type of distance to us and which attribute to use to product the best results.
4. Computation cost is quite high because we need to compute distance of each two points
http://people.revoledu.com/kardi/tutorial/KNN/Strength%20and%20Weakness.htm

2. Decision Tree:
Algorithm:



```
(1)  create a node $N$;
(2)  if tuples in $D$ are all of the same class, $C$ then
(3)      return $N$ as a leaf node labeled with the class $C$;
(4)  if attribute_list is empty then
(5)      return $N$ as a leaf node labeled with the majority class in $D$; // majority voting
(6)  apply Attribute_selection_method($D$, attribute_list) to find the "best" splitting_criterion;
(7)  label node $N$ with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list – splitting_attribute; // remove splitting_attribute
(10) for each outcome $j$ of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let $D_j$ be the set of data tuples in $D$ satisfying the outcome $j$; // a partition
(12)     if $D_j$ is empty then
(13)         attach a leaf labeled with the majority class in $D$ to node $N$;
(14)     else attach the node returned by Generate_decision_tree($D_j$, attribute_list) to node $N$;
     endfor
(15) return $N$;
```

http://www.csun.edu/~twang/595DM/Slides/Week4.pdf

One example of our tree looks like below:

```
---------------------------------start printing tree---------------------------------
feature 22 >= 115.7?
>> True:
        feature 24 >= 0.08822?
        >> True:
                feature 22 >= 117.7?
                >> True:
                        Leaf with label of '1'
                >> False:
                        feature 21 >= 29.87?
                        >> True:
                                Leaf with label of '1'
                        >> False:
                                Leaf with label of '0'
        >> False:
                Leaf with label of '0'
>> False:
        feature 27 >= 0.1459?
        >> True:
                feature 21 >= 25.84?
                >> True:
                        feature 4 >= 0.092?
                        >> True:
                                Leaf with label of '1'
                        >> False:
                                Leaf with label of '0'
                >> False:
                        feature 29 >= 0.1405?
                        >> True:
                                Leaf with label of '1'
                        >> False:
                                Leaf with label of '0'
        >> False:
                feature 23 >= 967.0?
                >> True:
                        Leaf with label of '1'
                >> False:
                        feature 27 >= 0.1112?
                        >> True:
                                feature 21 >= 34.27?
                                >> True:
                                        Leaf with label of '1'
                                >> False:
                                        Leaf with label of '0'
                        >> False:
                                feature 29 >= 0.05521?
                                >> True:
                                        Leaf with label of '0'
                                >> False:
                                        Leaf with label of '1'
---------------------------------end printing ---------------------------------
```

Parameter setting:

Depth	project3_dataset1				
	Accuracy	Precision	Recall	F-1	Running time
2	0.9261	0.9079	0.8890	0.8966	73.83s
3	0.9313	0.9351	0.8682	0.8987	97.44s

4	0.9331	0.9179	0.8987	0.9067	106.53s
5	0.9244	0.9067	0.8803	0.8920	120.79s
6	0.9209	0.8898	0.8938	0.8895	140.72s
7	0.9244	0.8945	0.8988	0.8947	142.89s
9	0.9209	0.8898	0.8938	0.8895	139.76s
11	0.9209	0.8898	0.8938	0.8895	139.04s
Infinity	0.9209	0.8898	0.8938	0.8895	143.05s

Depth: 4 is chosen as the depth of the decision tree.
The higher depth actually not only results in longer calculating time, also cause overfitting, which can be reflected from the decreased accuracy. And when the depth gets too small, like 2, or 3, the predictive power is relatively lower. Because the information of the input data is not exhaustively explored.

Numerical attribute & Categorical attribute:
We applied true or false conditions on both numerical data and categorical data:
For numerical data, the true condition is A is greater than or equal to B, the false condition is A is less than B.
For categorical data, the true condition is A is exactly the same as B, otherwise the condition is false.

How to Address Overfitting:
Post-pruning

1. Grow decision tree to its entirety
2. Trim the nodes of the decision tree in a bottom-up fashion
3. If generalization error improves after trimming, replace sub-tree by a leaf node.
4. Class label of leaf node is determined from majority class of instances in the sub-tree

Results:
From table1 we can see:

- Decision Tree has the worst performance on both dataset1 and dataset2 in terms of the accuracy and the efficiency.

Cross Validation Result of dataset1:

```
-----------------------------------------round: 0---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9565, f-1 score = 0.9778
-----------------------------------------round: 1---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9412, f-1 score = 0.9697
-----------------------------------------round: 2---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9286, f-1 score = 0.9630
```

```
-----------------------------------------round: 3---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9524, f-1 score = 0.9756
-----------------------------------------round: 4---------------------------------------------
accurcay = 0.9649, precision = 1.0000, recall = 0.9000, f-1 score = 0.9474
-----------------------------------------round: 5---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9630, f-1 score = 0.9811
-----------------------------------------round: 6---------------------------------------------
accurcay = 1.0000, precision = 1.0000, recall = 1.0000, f-1 score = 1.0000
-----------------------------------------round: 7---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9333, f-1 score = 0.9655
-----------------------------------------round: 8---------------------------------------------
accurcay = 0.9123, precision = 1.0000, recall = 0.8387, f-1 score = 0.9123
-----------------------------------------round: 9---------------------------------------------
accurcay = 0.9643, precision = 0.9545, recall = 0.9545, f-1 score = 0.9545

result:
accurcay = 0.9736, precision = 0.9955, recall = 0.9368, f-1 score = 0.9647

--- 31.19899606704712 seconds ---
```

Cross Validation Result of dataset2:

```
-----------------------------------------round: 0---------------------------------------------
accurcay = 0.7234, precision = 0.8333, recall = 0.4762, f-1 score = 0.6061
-----------------------------------------round: 1---------------------------------------------
accurcay = 0.7660, precision = 0.6000, recall = 0.4615, f-1 score = 0.5217
-----------------------------------------round: 2---------------------------------------------
accurcay = 0.6739, precision = 0.7000, recall = 0.3684, f-1 score = 0.4828
-----------------------------------------round: 3---------------------------------------------
accurcay = 0.6522, precision = 0.4545, recall = 0.3333, f-1 score = 0.3846
-----------------------------------------round: 4---------------------------------------------
accurcay = 0.6739, precision = 0.8571, recall = 0.3000, f-1 score = 0.4444
-----------------------------------------round: 5---------------------------------------------
accurcay = 0.6087, precision = 0.5000, recall = 0.1667, f-1 score = 0.2500
-----------------------------------------round: 6---------------------------------------------
accurcay = 0.7826, precision = 0.7143, recall = 0.3846, f-1 score = 0.5000
-----------------------------------------round: 7---------------------------------------------
accurcay = 0.8043, precision = 0.6250, recall = 0.4545, f-1 score = 0.5263
-----------------------------------------round: 8---------------------------------------------
accurcay = 0.5652, precision = 0.3000, recall = 0.1875, f-1 score = 0.2308
-----------------------------------------round: 9---------------------------------------------
accurcay = 0.7826, precision = 0.8333, recall = 0.3571, f-1 score = 0.5000

result:
accurcay = 0.7033, precision = 0.6418, recall = 0.3490, f-1 score = 0.4447
```

```
--- 10.264924049377441 seconds ---
```

Conclusions:
Pros:
1. Inexpensive to construct
2. Extremely fast at classifying unknown records
3. Easy to interpret for small-sized trees
4. Accuracy is comparable to other classification techniques for many simple data sets

Adopted from Professor Jing Gao's PPT

Cons:
1. Decision Trees do not work well if you have smooth boundaries.
2. Decision Tree's do not work best if you have a lot of un-correlated variables.
3. Data fragmentation : Each split in a tree leads to a reduced dataset under consideration. And, hence the model created at the split will potentially introduce bias.
4. High variance and unstable : As a result of the greedy strategy applied by decision tree's variance in finding the right starting point of the tree can greatly impact the final result.

https://www.quora.com/What-are-the-disadvantages-of-using-a-decision-tree-for-classification

3. Naïve Bayes:

Algorithm:

Naive bayes is a conditional probability model: given a dataset with multiple features, calculate the probability for each possible class this dataset belongs to and select the highest probability as the predicted class.

The probability function is:

$$\boxed{P(H_i \mid X) = \frac{P(H_i)P(X \mid H_i)}{P(X)}}$$

We assume the value of particular feature is independent of any other features, so P(X|Hi) can be the product of multiple Descriptor Posterior Probability for each feature. For each possible classes Hi, we calculate the probability P(Hi|X) and pick the class with highest probability as our predicted result.

Numerical attribute & Categorical attribute:

We need to handle the case for Numerical and Categorical attribute in different ways. For Categorical attribute, it is easy to calculate the Descriptor Posterior Probability. For Numerical attribute, we assume it is a normal distribution, so we can use the probability density function value.

Avoiding the Zero-Probability Problem
Use Laplacian correction (or Laplacian estimator): add 1 to each case

Results:
From table1 we can see:

- Naive Bayes doesn't have a very strong predictive power. It is only a little stronger than Decision Tree
- One of its advantage is that it is the fastest algorithm among all of these algorithms. Because it doesn't need time to construct a predictive model and it also doesn't need heavy calculations.

```
Cross Validation Result of dataset1:
------------------------------------------round: 0------------------------------------------
accurcay = 0.9474, precision = 0.9167, recall = 0.9565, f-1 score = 0.9362
------------------------------------------round: 1------------------------------------------
accurcay = 0.9298, precision = 0.8824, recall = 0.8824, f-1 score = 0.8824
------------------------------------------round: 2------------------------------------------
accurcay = 0.9649, precision = 0.8750, recall = 1.0000, f-1 score = 0.9333
------------------------------------------round: 3------------------------------------------
accurcay = 0.9123, precision = 0.9000, recall = 0.8571, f-1 score = 0.8780
------------------------------------------round: 4------------------------------------------
accurcay = 0.8947, precision = 0.8889, recall = 0.8000, f-1 score = 0.8421
------------------------------------------round: 5------------------------------------------
accurcay = 0.9649, precision = 1.0000, recall = 0.9259, f-1 score = 0.9615
------------------------------------------round: 6------------------------------------------
accurcay = 0.9649, precision = 0.9545, recall = 0.9545, f-1 score = 0.9545
------------------------------------------round: 7------------------------------------------
accurcay = 0.9649, precision = 0.9333, recall = 0.9333, f-1 score = 0.9333
------------------------------------------round: 8------------------------------------------
accurcay = 0.9123, precision = 0.9643, recall = 0.8710, f-1 score = 0.9153
------------------------------------------round: 9------------------------------------------
accurcay = 0.8929, precision = 0.8636, recall = 0.8636, f-1 score = 0.8636

result:
accurcay = 0.9349, precision = 0.9179, recall = 0.9044, f-1 score = 0.9100

--- 18.508169174194336 seconds ---

Cross Validation Result of dataset2:
------------------------------------------round: 0------------------------------------------
accurcay = 0.6596, precision = 0.6000, recall = 0.7143, f-1 score = 0.6522
------------------------------------------round: 1------------------------------------------
accurcay = 0.7447, precision = 0.5238, recall = 0.8462, f-1 score = 0.6471
------------------------------------------round: 2------------------------------------------
accurcay = 0.7826, precision = 0.7647, recall = 0.6842, f-1 score = 0.7222
```

```
------------------------------------------round: 3----------------------------------------------
accurcay = 0.6739, precision = 0.5000, recall = 0.6000, f-1 score = 0.5455
------------------------------------------round: 4----------------------------------------------
accurcay = 0.6304, precision = 0.6000, recall = 0.4500, f-1 score = 0.5143
------------------------------------------round: 5----------------------------------------------
accurcay = 0.6304, precision = 0.5263, recall = 0.5556, f-1 score = 0.5405
------------------------------------------round: 6----------------------------------------------
accurcay = 0.8478, precision = 0.7500, recall = 0.6923, f-1 score = 0.7200
------------------------------------------round: 7----------------------------------------------
accurcay = 0.7609, precision = 0.5000, recall = 0.5455, f-1 score = 0.5217
------------------------------------------round: 8----------------------------------------------
accurcay = 0.6087, precision = 0.4444, recall = 0.5000, f-1 score = 0.4706
------------------------------------------round: 9----------------------------------------------
accurcay = 0.6957, precision = 0.5000, recall = 0.5714, f-1 score = 0.5333

result:
accurcay = 0.7035, precision = 0.5709, recall = 0.6159, f-1 score = 0.5867

--- 3.3338112831115723 seconds ---
```

Conclusions:
Pros:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.
- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Source: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

4. Random Forests:

Algorithm:

Each tree is constructed using the following algorithm:

1. Let the number of training cases be N, and the number of variables in the classifier be M.
2. We are told the number m of input variables to be used to determine the decision at a node of the tree; m should be much less than M.
3. Choose a training set for this tree by choosing n times with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
4. For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.
5. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.
http://amateurdatascientist.blogspot.com/2012/01/random-forest-algorithm.html

Parameter setting:
Feature numbers: Typically, for a classification problem with p features, √p (rounded down) features are used in each split.
https://en.wikipedia.org/wiki/Random_forest
The other two parameters are sampe number and tree number, which can be determined by the following analysis:

sample num	project3_dataset1, tree number = 100				
	Accuracy	Precision	Recall	F-1	Running time
len(train_set)	0.9402	0.9240	0.9124	0.9173	119.86s
len(train_set) /2	0.9524	0.9607	0.9033	0.9305	36.06s
len(train_set) /5	0.9438	0.9702	0.8760	0.9193	5.967s
len(train_set) /10	0.9367	0.9667	0.8620	0.9109	1.596s

len(train_set) /20	0.9121	0.9594	0.8015	0.8707	0.408s
len(train_set) /50	0.8998	0.9426	0.7832	0.8494	0.087s
len(train_set) /100	0.8068	0.9847	0.4874	0.6199	0.0356s

From this table, we chose len(train_set) / 5 as the length of sample size, because it achieves very good balance between running time and accuracy.

tree number	project3_dataset1, sample num = len(train_set) / 5				
	Accuracy	Precision	Recall	F-1	Running time
1	0.9033	0.8796	0.8534	0.8634	0.5818s
10	0.9402	0.9521	0.8856	0.9162	6.5010s
30	0.9596	0.9745	0.9199	0.9457	18.516s
50	0.9613	0.9896	0.9038	0.9440	31.015s
70	0.9472	0.9731	0.8808	0.9241	43.809s
100	0.9454	0.9668	0.8871	0.9242	60.709s
200	0.9209	0.8898	0.8938	0.8895	139.76s

Because we do bagging and randomly pick a certain number of features for generating the forests, so every time these scores are slightly different. But still we can see when the tree number is increased to 30 or 50, the predictive power reaches the peak power. So we'll chose 50 as our tree number.

How to get overfitting for each tree:
We stop when one of the following happens:

- A node becomes too small (<= 3 examples).
- the total height of the tree exceeds some limits, such as the total number of features.

Results:
From table1 we can see:

- Random forests has better predict results compared with decision tree. That's because it can get rid of bias and it doesn't have overfitting, which often occurs in decision tree model.
- Because the depth of the tree is much shorter than regular decision tree and also it use bagging to resample, random forests run much faster than the decision tree.

```
Cross Validation Result of dataset1:
-------------------------------------------round: 0---------------------------------------------
accurcay = 0.9825, precision = 0.9583, recall = 1.0000, f-1 score = 0.9787
-------------------------------------------round: 1---------------------------------------------
accurcay = 0.9474, precision = 0.9375, recall = 0.8824, f-1 score = 0.9091
-------------------------------------------round: 2---------------------------------------------
accurcay = 0.9649, precision = 1.0000, recall = 0.8571, f-1 score = 0.9231
-------------------------------------------round: 3---------------------------------------------
accurcay = 0.9649, precision = 1.0000, recall = 0.9048, f-1 score = 0.9500
-------------------------------------------round: 4---------------------------------------------
accurcay = 0.9474, precision = 1.0000, recall = 0.8500, f-1 score = 0.9189
-------------------------------------------round: 5---------------------------------------------
accurcay = 0.9474, precision = 1.0000, recall = 0.8889, f-1 score = 0.9412
-------------------------------------------round: 6---------------------------------------------
accurcay = 0.9649, precision = 1.0000, recall = 0.9091, f-1 score = 0.9524
-------------------------------------------round: 7---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9333, f-1 score = 0.9655
-------------------------------------------round: 8---------------------------------------------
accurcay = 0.9474, precision = 1.0000, recall = 0.9032, f-1 score = 0.9492
-------------------------------------------round: 9---------------------------------------------
accurcay = 0.9643, precision = 1.0000, recall = 0.9091, f-1 score = 0.9524

result:
accurcay = 0.9613, precision = 0.9896, recall = 0.9038, f-1 score = 0.9440

--- 32.384408950805664 seconds ---


Cross Validation Result of dataset2:
-------------------------------------------round: 0---------------------------------------------
accurcay = 0.7234, precision = 0.8333, recall = 0.4762, f-1 score = 0.6061
-------------------------------------------round: 1---------------------------------------------
accurcay = 0.7660, precision = 0.6000, recall = 0.4615, f-1 score = 0.5217
-------------------------------------------round: 2---------------------------------------------
accurcay = 0.6739, precision = 0.7000, recall = 0.3684, f-1 score = 0.4828
-------------------------------------------round: 3---------------------------------------------
accurcay = 0.6522, precision = 0.4545, recall = 0.3333, f-1 score = 0.3846
-------------------------------------------round: 4---------------------------------------------
accurcay = 0.6739, precision = 0.8571, recall = 0.3000, f-1 score = 0.4444
```

```
------------------------------------------round: 5----------------------------------------------
accurcay = 0.6087, precision = 0.5000, recall = 0.1667, f-1 score = 0.2500
------------------------------------------round: 6----------------------------------------------
accurcay = 0.7826, precision = 0.7143, recall = 0.3846, f-1 score = 0.5000
------------------------------------------round: 7----------------------------------------------
accurcay = 0.8043, precision = 0.6250, recall = 0.4545, f-1 score = 0.5263
------------------------------------------round: 8----------------------------------------------
accurcay = 0.5652, precision = 0.3000, recall = 0.1875, f-1 score = 0.2308
------------------------------------------round: 9----------------------------------------------
accurcay = 0.7826, precision = 0.8333, recall = 0.3571, f-1 score = 0.5000

result:
accurcay = 0.7033, precision = 0.6418, recall = 0.3490, f-1 score = 0.4447

--- 10.264924049377441 seconds ---
```

Conclusions:
Pros:

- It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It runs efficiently on large databases.
- 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.
- It has methods for balancing error in class population unbalanced data sets.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

Cons:

- Random forests have been observed to overfit for some datasets with noisy classification/regression tasks

http://amateurdatascientist.blogspot.com/2012/01/random-forest-algorithm.html

5. Adaboost Decision Tree:

Algorithm:

After a tree is trained, we can compare the training sample scores sisi with the training sample true identities yiyi. We define a function to indicate whether an event is classified incorrectly: I(s,y) = 0 if s = y and 1 otherwise. Then we calculate the error rate for the tree:

$$e = \frac{\sum_i w_i I(s_i, y_i)}{\sum_i w_i}$$

and the boost factor for the tree:

$$\alpha = \beta \cdot \ln\left(\frac{1-e}{e}\right).$$

Here, β is a user-specified boost strength (typically between 0 and 1). Once we have the boost factor for the tree, we adjust the weights:

$$w_i \to w_i \cdot \exp[\alpha \cdot I(s_i, y_i)].$$

Finally, the weights are renormalized so that ∑iwi=1. The new weights are used to train the next tree. After that, the weights are boosted yet again. Boosting is cumulative; the weights are never reset to their original values.
Calculating the BDT score
Consider a set of trees with indices mm. The boost factor αmαm calculated during boosting becomes the weight of that tree during scoring. When not using leaf purity information, the score of an event is

$$s_i = \frac{\sum_m \alpha_m (s_i)_m}{\sum_m \alpha_m}.$$

http://software.icecube.wisc.edu/documentation/projects/pybdt/man_bdt_intro.html

Parameter setting:
The two parameters are sampe number and tree number, which can be determined by the following analysis:

sample num	project3_dataset1, tree number = 10				
	Accuracy	Precision	Recall	F-1	Running time
len(train_set)	0.9419	0.9361	0.9077	0.9203	155.98s

len(train_set) /2	0.9367	0.9297	0.9048	0.9146	51.42s
len(train_set) /5	0.9508	0.9372	0.9276	0.9312	11.45s
len(train_set) /10	0.9455	0.9536	0.9015	0.9245	3.65s
len(train_set) /20	0.9560	0.9692	0.9122	0.9387	1.297s
len(train_set) /50	0.9455	0.9391	0.9090	0.9228	0.4722s
len(train_set) /100	0.9016	0.9000	0.8482	0.8658	0.2864s

From this table, we can see the sample size doesn't affect the predictive power to much when the size is more than 1/50 of the original size. we chose len(train_set) / 20 as the length of sample size, because it achieves very good balance between running time and accuracy.

tree number	project3_dataset1, sample num = len(train_set) / 5				
	Accuracy	Precision	Recall	F-1	Running time
10	0.9578	0.9536	0.9341	0.9420	1.345s
30	0.9630	0.9616	0.9444	0.9508	3.50s
50	0.9684	0.9641	0.9559	0.9580	6.058s
70	0.9648	0.9739	0.9270	0.9491	8.040s
100	0.9789	0.9907	0.9539	0.9714	11.291s
200	0.9789	0.9957	0.9511	0.9722	23.31s
500	0.9754	0.9957	0.9414	0.9670	56.86s
1000	0.9754	0.9858	0.9507	0.9670	119.38s

We can see when the tree number is increased to more than 100, the predictive power reaches the peak power. So we'll chose 50 as our tree number.

How to get overfitting for each tree:
The depth of each tree is at most 2, which means the tree actually is a stump.

Results:
From table1 we can see:

- Adaboost decision tree shows the best performance in terms of only accuracy among all of these five algorithms.
- Adaboost decision tree is very efficient. This is because everytime we only extract a certain number of samples and features to build the tree, which can highly shorter the calculation time to get the best split.

Cross Validation Result of dataset1:
```
-------------------------------------------round: 0---------------------------------------------
accurcay = 1.0000, precision = 1.0000, recall = 1.0000, f-1 score = 1.0000
-------------------------------------------round: 1---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9412, f-1 score = 0.9697
-------------------------------------------round: 2---------------------------------------------
accurcay = 0.9649, precision = 1.0000, recall = 0.8571, f-1 score = 0.9231
-------------------------------------------round: 3---------------------------------------------
accurcay = 0.9649, precision = 0.9524, recall = 0.9524, f-1 score = 0.9524
-------------------------------------------round: 4---------------------------------------------
accurcay = 0.9649, precision = 0.9500, recall = 0.9500, f-1 score = 0.9500
-------------------------------------------round: 5---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9630, f-1 score = 0.9811
-------------------------------------------round: 6---------------------------------------------
accurcay = 1.0000, precision = 1.0000, recall = 1.0000, f-1 score = 1.0000
-------------------------------------------round: 7---------------------------------------------
accurcay = 0.9825, precision = 1.0000, recall = 0.9333, f-1 score = 0.9655
-------------------------------------------round: 8---------------------------------------------
accurcay = 0.9298, precision = 1.0000, recall = 0.8710, f-1 score = 0.9310
-------------------------------------------round: 9---------------------------------------------
accurcay = 0.9821, precision = 0.9565, recall = 1.0000, f-1 score = 0.9778

result:
accurcay = 0.9754, precision = 0.9859, recall = 0.9468, f-1 score = 0.9651

--- 22.20963191986084 seconds ---
```

Cross Validation Result of dataset2:
```
-------------------------------------------round: 0---------------------------------------------
accurcay = 0.7234, precision = 0.7000, recall = 0.6667, f-1 score = 0.6829
```

```
-----------------------------------------round: 1---------------------------------------------
accurcay = 0.7447, precision = 0.5263, recall = 0.7692, f-1 score = 0.6250
-----------------------------------------round: 2---------------------------------------------
accurcay = 0.7826, precision = 0.8000, recall = 0.6316, f-1 score = 0.7059
-----------------------------------------round: 3---------------------------------------------
accurcay = 0.6522, precision = 0.4667, recall = 0.4667, f-1 score = 0.4667
-----------------------------------------round: 4---------------------------------------------
accurcay = 0.6304, precision = 0.7143, recall = 0.2500, f-1 score = 0.3704
-----------------------------------------round: 5---------------------------------------------
accurcay = 0.6739, precision = 0.6364, recall = 0.3889, f-1 score = 0.4828
-----------------------------------------round: 6---------------------------------------------
accurcay = 0.6739, precision = 0.4167, recall = 0.3846, f-1 score = 0.4000
-----------------------------------------round: 7---------------------------------------------
accurcay = 0.8043, precision = 0.5833, recall = 0.6364, f-1 score = 0.6087
-----------------------------------------round: 8---------------------------------------------
accurcay = 0.6087, precision = 0.4167, recall = 0.3125, f-1 score = 0.3571
-----------------------------------------round: 9---------------------------------------------
accurcay = 0.6739, precision = 0.4545, recall = 0.3571, f-1 score = 0.4000

result:
accurcay = 0.6968, precision = 0.5715, recall = 0.4864, f-1 score = 0.5099

--- 7.665283203125 seconds ---
```

Conclusions:
Pros:

- It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It runs efficiently on large databases.
- 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.

Cons:

- Sensitive to noise and outliers
- Not easy to implement