A COMPARATIVE STUDY OF BAGGING AND
BOOSTING OF SUPERVISED AND
UNSUPERVISED CLASSIFIERS FOR
OUTLIERS DETECTION

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering

by

YUE DANG
B.E, University of Electronic Science and Technology of China, 2011

2017
Wright State University
Wright State University
Graduate School

July 10, 2017

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Yue Dang ENTITLED A Comparative Study of Bagging and Boosting of Supervised and Unsupervised Classifiers For Outliers Detection BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Electrical Engineering.

Zhiqiang Wu, Ph.D.
thesis Director

Brian Rigling, Ph.D.
Department Chair

Committee on Final Examination

Zhiqiang Wu, Ph.D.

Yan Zhuang, Ph.D.

Bin Wang, Ph.D.

Robert E. W. Fyffe, Ph.D.
Vice President for Research and Dean of the Graduate School
ABSTRACT

Dang, Yue. M.S.E.E., Department of Electrical Engineering, Wright State University, 2017. A Comparative Study of Bagging and Boosting of Supervised and Unsupervised Classifiers For Outliers Detection.

The problem of outlier detection has received increasing attention recently because it plays a great role in many fields such as credit fraud detection, cyber security, etc. Machine Learning approach is an excellent choice for outlier detection due to its accuracy and efficiency. Outlier detection problem is unique due to the so-called classes imbalance: the inliers are extreme majority and the outliers are minority. Ensemble methods are popular in classification and regression task in practice to improve the performance of machine learning algorithms. Bagging and boosting are two common methods of them. In this thesis, we want to show the performance of bagging and boosting compared with base algorithms in outlier detection. First of all, some basic algorithms for outlier detection are described for both supervised and unsupervised methods. Next, theoretical analysis and strategies of ensemble are discussed. Furthermore, groups of experiments are conducted and the experiment results confirm the effectiveness of bagging and boosting methods for outlier detection problem.
## Contents

1 Chapter 1: Introduction .......................................................... 1  
1.1 Problem Description ......................................................... 1  
1.2 Motivation ........................................................................ 2  
1.3 Thesis Outline ................................................................. 2  

2 Chapter 2: Basic Algorithms .................................................... 4  
2.1 Supervised Approach ......................................................... 4  
  2.1.1 Cost-sensitive Decision Tree ......................................... 5  
  2.1.2 Cost-sensitive Support Vector Machine ........................... 7  
2.2 Unsupervised Approach ..................................................... 10  
  2.2.1 Distance Based Approaches ......................................... 10  
  2.2.2 Density Based Approach ............................................. 11  

3 Chapter 3: Ensemble Learning ................................................ 14  
3.1 Frame of Ensemble Learning .............................................. 14  
3.2 Bagging and Boosting ....................................................... 15  
  3.2.1 Bagging ................................................................. 15  
  3.2.2 Boosting ............................................................... 16  
3.3 Bias-Variance Tradeoff ...................................................... 18  
3.4 Strategy of diversity .......................................................... 21  
  3.4.1 Training data perturbation ......................................... 21  
  3.4.2 Feature perturbation ............................................... 22  
  3.4.3 Parameters Perturbation .......................................... 22  

4 Chapter 4: Experiments and Results ...................................... 23  
4.1 Evaluation Method ............................................................ 23  
4.2 Data set Description and Experiment Environment ............... 26  
4.3 Experiments setup and results ......................................... 27  
  4.3.1 Supervised Ensemble ............................................... 27  
  4.3.2 Unsupervised Ensemble .......................................... 31  

5 Conclusion ........................................................................... 42
List of Figures

3.1 The frame of bagging ........................................... 16
3.2 The frame of boosting ........................................... 17
3.3 bias-variance dilemma ........................................... 20
4.1 Precision-Recall Curve .......................................... 24
4.2 ROC Curve ......................................................... 25
4.3 supervised results of cardio dataset .......................... 28
4.4 supervised results of pageblock dataset ....................... 29
4.5 supervised results of wilt dataset ............................. 29
4.6 supervised results of thyroid dataset ......................... 30
4.7 unsupervised results of cardio dataset ....................... 32
4.7 unsupervised results of cardio dataset continued .......... 33
4.8 unsupervised results of pageblock dataset ................... 34
4.8 unsupervised results of pageblock dataset continued ...... 35
4.9 unsupervised results of wilt dataset ........................ 36
4.9 unsupervised results of wilt dataset continued ............ 37
4.10 unsupervised results of thyroid dataset ..................... 38
4.10 unsupervised results of thyroid dataset continued ........ 39
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>confusion matrix</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>summary of data sets</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>software and hardware environment of experiments</td>
<td>27</td>
</tr>
<tr>
<td>4.4</td>
<td>supervised ensemble results</td>
<td>30</td>
</tr>
<tr>
<td>4.5</td>
<td>unsupervised ensemble results based on knn</td>
<td>40</td>
</tr>
<tr>
<td>4.6</td>
<td>unsupervised ensemble results based on knnw</td>
<td>40</td>
</tr>
<tr>
<td>4.7</td>
<td>unsupervised ensemble results based on odin</td>
<td>40</td>
</tr>
<tr>
<td>4.8</td>
<td>unsupervised ensemble results based on lof</td>
<td>41</td>
</tr>
<tr>
<td>4.9</td>
<td>unsupervised ensemble results based on slof</td>
<td>41</td>
</tr>
<tr>
<td>4.10</td>
<td>unsupervised ensemble results based on inflo</td>
<td>41</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

1.1 Problem Description

The problem of outlier detection has recently received increasing attention because it plays an important role in many fields such as credit fraud detection, cyber security, etc. The purpose of outlier detection is to identify the items which do not conform to other items in a data set. For example, for some disease diagnosis, most people are not carrying the disease and only a very few are real patients. In another example of cyber security applications, most connections are normal and just a few of them are under attack. For credit card records, most transactions are normal and only a very small portion of them are fraud ones. Although these outliers (e.g., the patients, the attacked connections and the fraud transactions) occur rarely, they are very important and meaningful to the users, and they are what we want to detect in mass data.

Researchers have attempted many machine learning approaches to tackle this problem. From the view of machine learning task, outlier detection can be divided into two categories: supervised and unsupervised. If outlier and inlier labels exist in the training data set and the aim is to obtain labels for samples in test data set, it’s a supervised problem. Different from general supervised problem, class imbalance is obvious in outlier detection: outliers are extreme minority and inliers are majority. If there are no labels, the task is to decide whether one item belongs to the same distribution as existing observations (an inlier), or should be considered as different (an outlier). Due to the class imbalance,
outlier detection faces some unique challenges.

### 1.2 Motivation

Numerous algorithms have been proposed for outlier detection in recent years [2] [3] [4] [7] [21] [20] [17] [19] [25] [26]. A detailed survey on the topic may be found in [8].

A given model may sometimes behave well on a given data set, but may not behave well on other data sets. Ensemble analysis is a method which is commonly used in order to reduce the dependence of the model on specific data set or data locality. This greatly increases the robustness of the model. The ensemble technique is popular in problems such as clustering and classification. Bagging [5], boosting [14], stacking [9] [29] [30], random forests [6], model averaging [11] and bucket of models [12] are typical ensemble methods. They are confirmed to have the ability to perform better than individual algorithms and will not take much more cost than individual learners.

For outlier detection, which is somewhat different from the classification and clustering problems referred in the section above, ensemble methods deserve attention as well. In this thesis, a comparative study of bagging and boosting of both supervised and unsupervised is implemented. The performances of these ensemble methods in various data sets compared with individual algorithms are meaningful and desirable for outlier detection problem.

### 1.3 Thesis Outline

The rest of the thesis is organized as follows: Chapter 1 provides a basic introduction of outlier detection problem. In Chapter 2, basic algorithms for both supervised and unsupervised outlier detection are described. In Chapter 3, ensemble methods are presented. Additionally, some theoretical analysis of the reason that ensemble improves performance
is given. Fundamental strategies are also discussed in detail. In Chapter 4, we evaluate the performances of ensemble method compared with corresponding basic algorithms and results are illustrated and analyzed.
Chapter 2: Basic Algorithms

2.1 Supervised Approach

If labels are attached to the training samples, the problem of outlier detection can be seen as a supervised problem. In this condition, it appears similar to a classification problem. The aim of outlier detection is to find outliers from all the data and the result is the label for each sample: inlier or outlier. From this perspective, this is similar with bi-classification. However, the difference is the proportion of the two classes. In a typical classification problem, the default hypothesis is that the number of negative and positive samples are close to each other. In outlier detection, however, the amount of the two classes are extremely imbalanced. In other words, supervised methods can be used but special attention should be made for imbalance adjustment.

There exist a few methods for imbalanced data with supervised methods: undersampling, oversampling and cost-sensitive learning. Undersampling means removing samples from the majority class using an undersampling algorithm. Oversampling means generating new samples from the minority class using an oversampling algorithm. Cost-sensitive learning assigns the misclassifications of minority class samples with a higher cost than misclassifications of majority class samples.
2.1.1 Cost-sensitive Decision Tree

Decision tree [22] is a commonly used algorithm for classification and regression. Decision tree for classification uses tree structure to classify the instances. The merit of decision tree is that it is highly readable.

As a tree structure, it consists of one root node, several internal nodes and several leaf nodes. Lead nodes determine results. Internal nodes are tests for feature values and instances in one node are allocated to its child nodes according to the feature test. Root node contains all the instances. Therefore, each path from root node to leaf node is a test sequence and the goal is to learn a decision tree that with great generalization ability which deals with new instances never seen before.

The loss function of decision tree is a maximum likelihood function with regularization. Selecting optimal tree from all possible trees is an NP problem. Therefore, heuristic strategy is always used to tackle this optimization problem to obtain a sub-optimal solution. The process of building a decision tree is a divide-and-conquer strategy, as Algorithm 1 shows.

The key step of the process is Step 8: select optimal feature to grow new branches. The guideline for selection is that the instances in the child node belong to the same class as much as possible, which is considered of high “purity”.

There are two common measures of impurity: Entropy and Gini.

If a target is a classification outcome taking on values $1, 2, \ldots, K$ of set $D$

$$p_k = \frac{N_k}{\sum_{k=1}^{K} N_k}$$  \hspace{1cm} (2.1)

The entropy of $D$ is defined as:

$$H(D) = -\sum_{k=1}^{K} p_k \log_2 p_k$$  \hspace{1cm} (2.2)
Algorithm 1 Building Decision Tree

**Input:** training set $D = \{(x_1, y_1), (x_2, y_2), ... (x_m, y_m)\}$; feature set $A = \{a_1, a_2, ..., a_d\}$

**Process:** Function TreeGenerate($D, A$)
1: Generate one node $node$;
2: if all the instances in $D$ belong to one class $C$ then
3: label $node$ as leaf node of $C$ class;
4: end if
5: if $A = \emptyset$ OR all the instances are same value in $A$ then
6: label $node$ as leaf node of the majority class in $D$;
7: end if
8: select optimal $a^*_s$ from $A$;
9: for each value $a^*_v$ in $a^*_s$ do
10: generate one branch for $node$; $D_v$ represents the subset of $a^*_v$;
11: if $D_v$ is empty then
12: label $node$ as leaf node of the majority class in $D$;
13: else
14: regard TreeGenerate($D_v, A \setminus \{a_s\}$) as internal node;
15: end if
16: end for

**Output:** a tree with $node$ as root

The Gini of $D$ corresponds to:

$$H(D) = 1 - \sum_{k=1}^{K} p_k^2 \quad (2.3)$$

Suppose $D$ will have left child node and right child node splitting by feature $a$, and $N^{left}$ represents the number of instances in left child node and $N^{right}$ represents the number of instances in right child node. Low value of child nodes means feature $a$ splitting resulting high purity increasing. Therefore,

$$G(D, a) = \frac{N^{left}}{N^{left} + N^{right}} H(D^{left}(a)) + \frac{N^{right}}{N^{left} + N^{right}} H(D^{right}(a)) \quad (2.4)$$
Selecting $a$ to minimize $G(D, a)$:

$$a^* = \text{argmin}_a G(D, a)$$  \hspace{1cm} (2.5)

For imbalanced condition, weights of classes are different. Class weights can be converted to instance weights. It is obvious that instance of outliers should be attached with higher weight and instance of inliers with lower weight.

$$W_{\text{outliers}} = \frac{N_{\text{outliers}} + N_{\text{inliers}}}{N_{\text{outliers}} \times N_{\text{classes}}}$$  \hspace{1cm} (2.6)

$$W_{\text{inliers}} = \frac{N_{\text{outliers}} + N_{\text{inliers}}}{N_{\text{inliers}} \times N_{\text{classes}}}$$  \hspace{1cm} (2.7)

Then, equation 2.1 should be modified as:

$$p_k = \frac{W_k N_k}{\sum_{k=1}^{K} W_k N_k}$$  \hspace{1cm} (2.8)

and equation 2.4 should be modified as:

$$G(D, a) = \sum_{k=1}^{K} \frac{N_{\text{left}}^k}{N_k} H(D_{\text{left}}^k(a)) + \sum_{k=1}^{K} \frac{N_{\text{right}}^k}{N_k} H(D_{\text{right}}^k(a))$$  \hspace{1cm} (2.9)

### 2.1.2 Cost-sensitive Support Vector Machine

The idea of Support Vector Machine [10] is to find a hyperplane to optimally separate instances of two classes. For training set $D = \{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}, y_i \in \{-1, +1\}$, in the sample space, the hyperplane can be described as:

$$w^T x = b$$  \hspace{1cm} (2.10)
where \( w \) is a normal vector, \( b \) is the intercept and the hyperplane is determined by \( w \) and \( b \).

Suppose the hyperplane can separate instances correctly, that is:

\[
\begin{cases}
  w^T x_i + b \geq +1, y_i = +1; \\
  w^T x_i + b \geq -1, y_i = -1.
\end{cases}
\]  

(2.11)

For some points that are the closest to hyperplane, the equality of the equation above is established. These points are called support vector. The sum distance between support vectors of two classes and hyperplane (called margin) is:

\[ \gamma = \frac{2}{||w||} \]  

(2.12)

To find the maximum margin:

\[
\max_{w, b} \frac{2}{||w||} \quad s.t. \ y_i(w^T x_i + b) \geq 1, i = 1, 2, ..., m.
\]  

(2.13)

It is equal to:

\[
\min_{w, b} \frac{1}{2}||w||^2 \quad s.t. \ y_i(w^T x_i + b) \geq 1, i = 1, 2, ..., m.
\]  

(2.14)

In most cases, the original sample space cannot be linearly separated. Samples should be mapped to another higher dimension in which samples can be linearly separated. Suppose \( \phi(x) \) is the mapped result from \( x \), the hyperplane is:

\[ f(x) = w^T \phi(x) + b \]  

(2.15)
Similar to equation 2.11:

\[
\min_{w,b} \frac{1}{2}||w||^2 \\
\text{s.t. } y_i(w^T \phi(x_i) + b) \geq 1, i = 1, 2, \ldots m.
\]

(2.16)

It is impossible to satisfy the constrains for all samples because of noisy data or other reasons. The point of “soft margin” is that several samples are allowed to not satisfy this constrain.

\[
\min_{w,b} \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \\
\text{s.t. } y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \\
\xi_i \geq 0, i = 1, 2, \ldots m.
\]

(2.17)

where \(C\) is the penalty factor which is a constant. \(C\) can balance the margin as large as possible and the number of misclassification instances as small as possible. When \(C\) is larger, the number of misclassification instances should be smaller; on the other hand, when \(C\) is smaller, more misclassification instances can be tolerated to gain larger margin. In extreme cases, when \(C\) is infinite, no misclassification is permitted, so it reduces to a “hard margin”; when \(C\) is zero, the latter item will be zero and no misclassification will be considered. For imbalanced condition, outliers should be paid extra attention. In general, inliers judged as outliers are preferred than outliers judged as inliers. Therefore, penalty factors for outliers and inliers should be different: the penalty factor for outliers set \(O\) should be larger and for inliers set \(I\) it should be smaller.
\[
\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C_1 \sum_{i: y_i \in \mathcal{O}} \xi_i + C_2 \sum_{i: y_i \in \mathcal{I}} \xi_i
\]

\[
s.t. \quad y_i (\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\
\xi_i \geq 0, \ i = 1, 2, \ldots m.
\]  

\textbf{2.2 Unsupervised Approach}

In unsupervised approach, we calculate outlier score for each instance. Outlier score is defined as the degree the instance being outlying. The higher this indicator value is, the value of the given item will be more of an outlier, and more likely to be a potential anomaly.

\textbf{2.2.1 Distance Based Approaches}

Distance Based Approaches are based on \( k \) nearest neighbours, which assign an outlier indicator value to each element based on its distance from its \( k \) nearest neighbours.

- \text{KNN}: Outlier score is the distance to \( k-\text{th} \) nearest neighbor.[26]

- \text{KNN-weight}: Outlier score is sum of the distances of the \( k \) nearest neighbors.[3]

- \text{ODIN(Outlier Detection using Indegree Number)].[16]

We define K-nearest neighbour graph as a weighted directed graph, in which every vertex represents a single point, and the edges correspond to pointers to neighbour points. In this graph, a point with less indegree number has higher possibility to be a outlier. Therefore, outlier score can be set to reciprocal of the indegree.
2.2.2 Density Based Approach

LOF (local outlier factor)[7]

The general concept of LOF is to compare the local density of a point with the densities of its neighbors. An outlier always has a much lower density than its neighbors. Some definitions are given to obtain the local outlier factor, which can be used as outlier score:

- **k-distance of an point p.**
  For any positive integer $k$, the $k$-distance of point $p$, denoted as $k$-distance$(p)$, is defined as the distance $d(p, o)$ between $p$ and an point $o \in D$ such that: (i) for at least $k$ objects $o \in D(o \neq p)$ it holds that $d(p, o) \leq d(p, o)$, and (ii) for at most $k - 1$ objects $o \in D(o \neq p)$ it holds that $d(p, o) < d(p, o)$

- **k-distance neighborhood of a point p**
  Given the $k$-distance of $p$, the $k$-distance neighborhood of $p$ contains every object whose distance from $p$ is not greater than the $k$-distance.

$$N_k(p) = q \in D(q \neq p) | d(p, q) \leq k \text{ - distance}(p)$$ (2.19)

These objects $q$ are called the $k$-nearest neighbors of $p$.

- **reachability distance**
  The reachability distance of point $p$ with respect to point $o$ is defined as

$$reach \text{ - dist}_k(p, o) = \max\{k \text{ - distance}(o), d(p, o)\}$$ (2.20)

Intuitively, if point $p$ is far away from $o$, the reachability distance between the two is simply their actual distance. However, if they are “sufficiently” close, the actual distance is replaced by the $k$-distance of $o$. 
• local reachability density

The local reachability density of $p$ is defined as:

$$lrd_k(p) = 1 / \left( \frac{\sum_{o \in N_k(p)^{reach}} \text{reach} - \text{dist}_k(p, o)}{|N_k(p)|} \right)$$ (2.21)

Intuitively, the local reachability density of $p$ is the inverse of the average reachability distance based on the $k$ neighbors of $p$. It is a density. If $p$ and its neighbors is in the same cluster, reach-distance would be small and the local reach density would be high. Inversely, if $p$ is far away with its neighbors, which means $p$ is a local outlier, reach-distance would be large and the local reach density would be low.

• local outlier factor

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}$$ (2.22)

It is the average of the ratio of the local reachability density of $p$ and those of $p$’s $k$-nearest neighbors. If the ratio is close to 1, i.e. the density of $p$ is close to that of its neighbors, then $p$ is possible to be in the same cluster with its neighbors. Inversely, if the ratio is higher than 1, i.e. the density of $p$ is lower than that of its neighbors, then $p$ is possible to be an local outlier.

LOF algorithm would perform well when aggregation degree in data set is different.

**SimplifiedLOF[28]**

Reach distance is set to be $k$-distance.

$$reach - dist_k(p, o) = k - distance(o)$$ (2.23)
INFLO[18]

Take both the nearest neighbors and reverse nearest neighbors into account, $NN_k(p)$ is a set of points which are $k$-nearest neighbors of $p$:

$$NN_k(p) = \{o \in D|o \neq p,d(p,o) \leq k-distance(p)\}$$

(2.24)

The density of $p$ is the reverse of the $k$-distance of $p$:

$$density(p) = \frac{1}{k-distance(p)}$$

(2.25)

$RNN_k(p)$ is the reverse $k$-nearest neighbors:

$$RNN_k(p) = \{q|q \in D,p \in NN_k(q)\}$$

(2.26)

By combining $NN_k(p)$ and $RNN_k(p)$ together in a novel way, we form a local neighborhood space which will be used to estimate the density distribution around $p$. We call this neighborhood space the $k$-influence space for $p$, denoted as $IS_k(p)$.

Then influenced outlierness is defined as:

$$INFLO_k(p) = \frac{\sum_{o \in IS_k(p)}density(o)}{|IS_k(p)|}$$

(2.27)

INFLO is the ratio of the average density of objects in $IS_k(p)$ to ps local density: ps INFLO will be very high if its density is much lower than those of its influence space objects. In this sense, $p$ will be an outlier.
Chapter 3: Ensemble Learning

3.1 Frame of Ensemble Learning

Ensemble learning is a process that combines multiple learners to solve a problem. The general frame of ensemble learning is illustrated as below: generate a group of individual learners, then combine them with some strategy. Individual learner is a basic algorithm trained with training data. In homogeneous ensemble method, individual learners are some basic algorithms (e.g., the components of “decision tree ensemble learner” are all decision trees). In homogeneous ensemble method, individual learners are called base learners. When it contains different type of basic algorithms, it is called heterogenous.

It is generally believed that ensemble learner always performs better than base learner. Here is a simple analysis to demonstrate this:

Consider binary classification problem, $y \in -1, +1$ and ground truth function $f$. Suppose the error rate of base learner is $\epsilon$, for each base learner $h_i$,

$$P(h_i(x) \neq f(x)) = \epsilon \quad (3.1)$$

Suppose ensemble learner combines $T$ base learners by a simple voting (if more than half of the base learners obtain the correct result, the result of the ensemble learner will be
correct):

\[ H(x) = \text{sign}(\sum_{i=1}^{T} h_i(x)) \]  

(3.2)

If base learners are independent with each other, according to Hoeffding inequalities [24], the error rate of ensemble learner is:

\[ P(H(x) \neq f(x)) = \sum_{k=0}^{T/2} \binom{T}{k} (1 - \epsilon)^k \epsilon^{(T-k)} \leq \left( -\frac{1}{2}T(1 - 2\epsilon)^2 \right) \]  

(3.3)

In the equation above, we can see that with the number of base learners \( T \) grows, the error rate of ensemble learner decreases.

However, there is a hypothesis in the analysis above which is that the error rate of base learners are independent with each other. In practice, it is obvious that base learners cannot be completely independent.

Based on the generating way of base learner, ensemble methods can be divided into two categories. In the first method, the latter base learners are strongly dependent on former ones and base learners are built sequentially. Boosting is the representative of this sequential approach. In the second method, there is no dependence with each base learners and they can be built in parallel. Bagging is a representative way of this parallel approach.

\section{3.2 Bagging and Boosting}

\subsection{3.2.1 Bagging}

Bagging [5] is the abbreviation of bootstrap aggregating. It is based on bootstrap [13] sampling: given a data set containing \( m \) samples, we take out \( m \) samples with replacement. Some instances appear in this sampling set many times and some instances don’t appear in it. We can do a simple estimate of one instance not appearing in the sampling set: in \( m \)
time sampling, the probability of one instance not taken out is $(1 - 1/m)^m$, the limit is:

$$\lim_{m \to \infty} (1 - \frac{1}{m})^m \approx 0.368$$  \hspace{1cm} (3.4)

Figure 3.1: The frame of bagging

Presently, $t$ sampling set can be generated, then $t$ learners can be trained based on each sampling set and then combined these $t$ learners to final result. All the above is the process of Bagging.

For unsupervised problem, bagging is easy to use because it doesn’t depend on intermediate evaluation in which step the label is necessary.

### 3.2.2 Boosting

Boosting [14] is an iterative process. After many iterations, the boosting algorithm combines weak learners to one strong learner. Iterative steps are as following:
Step 1: The base learner assigns equal weight or attention to each observation.

Step 2: According to the performance of former base learner, we pay higher attention to samples which did worse in former base learner. Then, we apply the next base learner.

Step 3: Iterate Step 2 till the number of learner reaches the setting in advance. Next, it combines the outputs from base learners in a certain way.

Figure 3.2: The frame of boosting

All the above is for supervised problem. For unsupervised problem, there is no label for internal validity measures in intermediate steps (Step 2). One commonly used heuristic approach, which is discussed in [1], is to remove outliers in successive iterations in order to build a successively more robust outlier model iteratively. This is a sequential ensemble. The basic idea is that outliers interfere with the creation of a model of normal data, and the removal of points with high outliers scores will be beneficial for the model in the next iteration. It is equivalent that instances with high outliers score in the former learner are assigned with the weight of zero for next iteration. It is not exactly the same as how boosting is understood in the supervised literature, but it is also a sequential method.
3.3 Bias-Variance Tradeoff

Bias-variance tradeoff is a good explanation of better performance of ensemble learners.

The performance of a learning machine on training data set is called “training error”, and the performance on test data set is called “generalization error”. The aim is to minimize the generalization error and ensemble is an important idea to minimize the generalization error. Bias-variance decomposition of generalization error is a good way to explain why ensemble approaches can perform better.

Bias-variance decomposition tries to decompose generalization error: Suppose $D$ is training data set. For test instance $x$, $y_D$ is donated as the label of $x$ in the training data set and $y$ is the real label of $x$. $f(x; D)$ is the output of model $f$ which is learned from $D$ of $x$.

The expected value of model $f$ in $x$ is:

$$\bar{f}(x) = \mathbb{E}_D[f(x; D)] \quad (3.5)$$

The variance due to different data $D$ is:

$$\text{var}(x) = \mathbb{E}_D[(f(x; D) - \bar{f}(x))^2], \quad (3.6)$$

The intrinsic noise is:

$$\epsilon^2 = \mathbb{E}_D[(y_D - y)^2] \quad (3.7)$$

The difference of expected output of the model and the real label is donated as bias:

$$\text{bias}^2(x) = (\bar{f}(x) - y)^2. \quad (3.8)$$

For simplicity, we can assume that the expected value of intrinsic noise is zero, that is
\[ E_D[y_D] - y = 0. \] Then, the expected generalization error can be decomposed as below:

\[
E(f; D) = E_D[(f(x; D) - y_D)^2] = E_D[(f(x; D) - \bar{f}(x) + \bar{f}(x) - y_D)^2] = E_D[((f(x; D) - \bar{f}(x))^2] + E_D[(\bar{f}(x) - y_D)^2] + E_D[2(f(x; D) - \bar{f}(x))(\bar{f}(x) - y)]
\]

\[
= E_D[((f(x; D) - \bar{f}(x))^2] + E_D[(\bar{f}(x) - y + y - y_D)^2] = E_D[((f(x; D) - \bar{f}(x))^2] + E_D[(\bar{f}(x) - y)^2] + E_D[(y - y_D)^2] + 2E_D[(\bar{f}(x) - y)(y - y_D)] = E_D[(f(x; D) - \bar{f}(x))^2] + (\bar{f}(x) - y)^2 + E_D[(y - y_D)^2]
\]

(3.9)

That is:

\[ E[(f; D)] = bias^2(x) + var(x) + \varepsilon^2 \]  

(3.10)

From the equation, we know that generalization error can be decomposed to bias, variance and intrinsic noise. Bias means the deviation between expected value of learning algorithm and real result. Variance means training data set varies results in the learning ability, that is the influence of data perturbation. Intrinsic noise is the lower bound of the generalization error of the current task, that is the difficulty of the task. The bias-variance decomposition shows that generalization error is jointly decided by the ability of algorithm, the sufficiency of the data and the difficulty of the task. Giving the task, to get the better performance of generalization, reducing bias and reducing variances are two approaches (the model fitting data as much as possible, and the influence of data perturbation as low as possible).

In general, conflicts always exist between reducing bias and variance. It is so called bias-variance dilemma, illustrated in the figure below 3.3. When training is insufficient, the ability of learning machine is not strong enough, data perturbation would not have significant influence on the learning machine and therefore bias leads to generalization.
error. With the degree of training grows, learning machine becomes stronger and data perturbation can be learned gradually, then variance is the dominant factor. After enough training, learning machine becomes strong enough and any tiny perturbation of training data will show effect on the learning machine. Overfitting would appear if the machine learns the non-global characteristic of training data.

Ensemble is a way that combines different models in order to ensure that the bias-variance tradeoff is optimized. This is achieved in two ways: reducing bias and reducing variance [15]. Bagging focuses on reducing variance: since each base learner generated in bagging is identically distributed (i.d.), the expectation of the average of $t$ base learners is the same as the expectation of any one of them:

$$E\left(\frac{1}{t} \sum_{i=1}^{t} X_i\right) = E(X_i)$$

(3.11)

So, it means bias will not be reduced. But variance can be changed. If $t$ variables are i.d.
(identically distributed) with positive pairwise correlation $\rho$, the variance of the average is:

$$Var\left(\frac{1}{t} \sum_{i=1}^{t} X_i\right) = \frac{1}{t^2} \{E[(\sum_{i=1}^{t} X_i)^2] - E[\sum_{i=1}^{t} X_i]^2]\} = \rho \sigma^2 + \frac{1 - \rho}{t} \sigma^2$$

(3.12)

If the variables are completely independent, then $\rho$ is zero and the first item in the equation is zero. Then, we can find that variance is reduced obviously. If the variables are same, then $\rho$ is one and the second item is zero. Then, the variance keeps constant. For bagging method, the training sets of each base learner are always overlapped but not the same, so $\rho$ is between zero and one, then the variance should be reduced. For boosting method, the base learners are strongly dependent, so $\rho$ is close to one, then variance will not be reduced obviously.

For unsupervised problem, there is no ground truth for training data, so the intrinsic error is missing. But unknown ground truth does exist. In the progress of aforementioned computation, there is no need for $f(x; D)$ to be computed. In conclusion, bias-variance tradeoff can generalize easily from supervised approaches to unsupervised approaches.

### 3.4 Strategy of diversity

The core of ensemble is how to generate “good and diverse” base learners. The general idea is to introduce randomness. The common method is creating some perturbation of training set, features, parameters of algorithms respectively.

#### 3.4.1 Training data perturbation

The main idea is to obtain different subsets from the initial training data, then build base learners with the subsets respectively. Sampling is the common method such as bootstrap
in bagging. This way is simple and effective for some algorithms which are sensitive to training data. For some stable base learner, which are not sensitive to training data, other perturbation should be introduced.

### 3.4.2 Feature Perturbation

Various subspace provides various views of training data and it is obvious that learners trained with different subspace will be diverse. For feature perturbation, subsets only contain some features of the initial training data.

For training data with redundant attributes, training base learners in subspace will not only produce diverse individual ones but also save time due to feature reducing. Under this condition, due to redundant attributes, base learners would not be too bad. But if initial training data contains too few features or redundant features are very few, feature perturbation is not a good choice.

### 3.4.3 Parameters Perturbation

There are always some parameters to be set for base learners, such as $k$ in K-Nearest-Neighbors algorithm. Various base learners can be trained with different parameters. It is worth noting that, when individual learners is trained, we always try several groups of parameters and then select the best ones. So, different learners have been trained already. It selects only the best one, but ensemble methods combine all of these learners together. Therefore, the costs of ensemble methods are not too larger than those of the individual learners.
Chapter 4: Experiments and Results

4.1 Evaluation Method

Outlier detection is a two-class problem, in which the outcomes are labeled either as positive (p) or negative (n). There are four possible outcomes. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP); however if the actual value is n then it is said to be a false positive (FP). Conversely, a true negative (TN) has occurred when both the prediction outcome and the actual value are n, and false negative (FN) is when the prediction outcome is n while the actual value is p. The four outcomes can be formulated in a $2 \times 2$ confusion matrix as follows.

Table 4.1: confusion matrix

<table>
<thead>
<tr>
<th></th>
<th>prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td></td>
</tr>
<tr>
<td></td>
<td>positive</td>
</tr>
<tr>
<td>true</td>
<td></td>
</tr>
<tr>
<td>positive</td>
<td>True Positive(TP)</td>
</tr>
<tr>
<td>negative</td>
<td>False Positive(FP)</td>
</tr>
</tbody>
</table>

The confusion matrix can derive several metrics. Precision and Recall are defined as follows:

\[
Precision = \frac{TP}{TP + FP} \quad (4.1)
\]

\[
Recall = \frac{TP}{TP + FN} \quad (4.2)
\]
Ideally, we hope $TP$ and $TN$ are the same with the number of true positive and negative instances, and then $FN$ and $FP$ would be zero. In practice, if we want to obtain more $TP$ through increasing the number of predict positive ones, recall will be high and precision will be low. On the other hand, if we just pick up the most likely instances, some true positive ones would be missing, then recall will be low and precision will be high. In other words, precision and recall are always in conflict: when precision becomes higher, recall would become lower and vice versa.

We can sort instances according to their outlier score (possibility). Set threshold of positive ones in this order one by one, then precision-recall curve can be obtained. For an ideal learning machine, precision can keep well with recall grows, as shown in the figure below. The bottom left area is the average precision, obviously the maximum is 1. The higher the area is, the better the learning machine is.

![Figure 4.1: Precision-Recall Curve](image)

The full name of ROC curve is Receiver Operating Characteristic. Similarly with obtaining Precision-Recall curve above, we sort all the instances according to outlier score,
then consider samples as positive ones one by one to get two values as vertical and horizontal coordinates. Different from Precision Recall curve, the vertical axis is True Positive Rate, and horizontal axis is False Positive Rate. They are defined as follows with signs in Figure 4.2:

\[ \text{True Positive Rate} = \frac{TP}{TP + FN}, \]

\[ \text{False Positive Rate} = \frac{FP}{TN + FP}. \]

Similar with Precision Recall curve, if ROC curve of one learner is completely “surrounded” by another curve of learner, we can determine that the performance of latter learner is better than that of the former learner. If two curves are crossed, the area under the curve (AUC) can then be the criterion (the larger, the better). For outlier detection, positive instances are few, so we pay more attention to the performance with low false positive rate. Here, the area under the curve and being the left part of false positive rate being 0.1 is considered, which is signed AUC0.1. Similarly, the larger the area is, the better the learner
4.2 Data set Description and Experiment Environment

We used four data sets from UCI machine learning repository [23]. A brief description is provided here:

The Cardiotocography (Cardio) data set contains measurements taken from fetal heart rate signals. The classes in the data set are normal, suspect, pathologic and the normal class forms the inliers while the pathologic class forms outlier class. The suspect class is discarded. The PageBlock data set are block information of documents. The data sets are divided into 5 classes: text, horizontal line, picture, vertical line and graphic. The text class forms inlier class and others form outlier class. The wilt is high-resolution remote sensing data set. Diseased trees class forms outlier class and the other land cover class forms inlier class. The thyroid data set contains data of thyroid disease. Similar with Cardio data, normal class forms the inliers while the hyperfunction class forms outlier class. Refer to table for details of data sets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#points</th>
<th>#dim</th>
<th>#outliers(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardio</td>
<td>1831</td>
<td>21</td>
<td>176 (9.6%)</td>
</tr>
<tr>
<td>PageBlocks</td>
<td>8473</td>
<td>10</td>
<td>560 (6.6%)</td>
</tr>
<tr>
<td>Wilt</td>
<td>4839</td>
<td>5</td>
<td>261 (5.3%)</td>
</tr>
<tr>
<td>Thyroid</td>
<td>7200</td>
<td>21</td>
<td>534 (7.4%)</td>
</tr>
</tbody>
</table>

The software and hardware environment of experiments are shown as 4.3.
Table 4.3: software and hardware environment of experiments

<table>
<thead>
<tr>
<th>software or hardware environment</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Inter Xeon E5-2560 @2.4GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>251G</td>
</tr>
<tr>
<td>Operating System</td>
<td>Ubuntu</td>
</tr>
<tr>
<td>Language</td>
<td>Python</td>
</tr>
<tr>
<td>Libraries</td>
<td>scikit-learn</td>
</tr>
</tbody>
</table>

4.3 Experiments setup and results

We set up groups of experiments to prove relative effectiveness of ensemble methods and bagging and boosting can improve performance of base learners in outlier detection problem.

4.3.1 Supervised Ensemble

Decision Tree and SVM algorithms are set as base learners.

For bagging method, 100 trials of base learners are used with 70 percent bootstrap samples. The outlier scores are also normalized in each trial then average of the 100 results is set as the ultimate score.

For boosting method, 100 iterations are used. Sample weights are changed in each iteration and the 100 learners are weighted according to their performance.

Stratified k fold cross validation are used. The proportion of outliers and inliers are kept constant in Stratified method [27].

From Figure 4.3, Figure 4.4, Figure 4.5, Figure 4.6 and the numbers in table, we can conclude that bagging and boosting methods perform better than their corresponding base algorithms in these four data sets. Bagging method sometimes gets results close to base learners, such as Figure 4.5(c) and Figure 4.6(d). From these two figures, we can see that base learners don’t perform well. According to theoretical analysis in above chapter, bagging method need relatively strong base learners. By contrast, boosting can be effective
Figure 4.3: supervised results of cardio dataset

(a) cardio DT ROC
(b) cardio DT PRC
(c) cardio SVM ROC
(d) cardio SVM PRC

with weak base learners.

The AUC value for ROC curve and average precision(AP) for each algorithm and each data set is shown in the table 4.4
Figure 4.4: supervised results of pageblock dataset

Figure 4.5: supervised results of wilt dataset
Figure 4.6: supervised results of thyroid dataset

Table 4.4: supervised ensemble results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cardio ROC AUC</th>
<th>Cardio AP</th>
<th>PageBlock ROC AUC</th>
<th>PageBlock AP</th>
<th>Wilt ROC AUC</th>
<th>Wilt AP</th>
<th>Thyroid ROC AUC</th>
<th>Thyroid AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT-base</td>
<td>0.70</td>
<td>0.73</td>
<td>0.91</td>
<td>0.88</td>
<td>0.74</td>
<td>0.43</td>
<td>0.99</td>
<td>0.97</td>
</tr>
<tr>
<td>DT-bagging</td>
<td>0.95</td>
<td>0.79</td>
<td>0.98</td>
<td>0.95</td>
<td>0.90</td>
<td>0.43</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>DT-boosting</td>
<td>0.98</td>
<td>0.91</td>
<td>0.99</td>
<td>0.96</td>
<td>0.98</td>
<td>0.78</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>SVM-base</td>
<td>0.89</td>
<td>0.57</td>
<td>0.91</td>
<td>0.63</td>
<td>0.75</td>
<td>0.11</td>
<td>0.87</td>
<td>0.54</td>
</tr>
<tr>
<td>SVM-bagging</td>
<td>0.91</td>
<td>0.65</td>
<td>0.99</td>
<td>0.90</td>
<td>0.75</td>
<td>0.10</td>
<td>0.86</td>
<td>0.49</td>
</tr>
<tr>
<td>SVM-boosting</td>
<td>0.95</td>
<td>0.80</td>
<td>0.95</td>
<td>0.90</td>
<td>0.82</td>
<td>0.16</td>
<td>0.87</td>
<td>0.58</td>
</tr>
</tbody>
</table>
4.3.2 Unsupervised Ensemble

Six algorithms referred in Section 2.2 are used as base learner. $k$ is set to 20.

For Bagging, three strategies of diversity (diversity of samples, diversity of features and diversity of model parameters which are referred in section 3.4) have been used. Specifically, they are bootstrap of 70 percent samples, bootstrap of 70 percent features and 30 $k$ values (from 10 to 300 with interval of 10). For Boosting, how it works in unsupervised problem is discussed in section 3.2. 0.5 percent samples which are suspected to be outliers would be discarded in each iteration. We use 50 iterations. Results for different data set are shown below.

Figures show the comparison of base algorithms and the three strategies of bagging and boosting. In Figure 4.7, bagging of samples and bagging of features perform almost the same with base learners, not outstanding. The small scale and limit of feature attributes of the data set may be the reason. Bagging of $k$ shows the obvious enhancement than base learners and boosting sometimes seems even better than bagging of $k$. In Figure 4.8, the observation is almost the same. In Figure 4.9 and Figure 4.10, precision and recall curves show that the base learner is weak, so bagging or boosting seems not improving the results. But in ROC curves, bagging and boosting methods also are the better ones.
Figure 4.7: unsupervised results of cardio dataset
Figure 4.7 (Cont.): unsupervised results of cardio dataset continued
Figure 4.8: unsupervised results of pageblock dataset
Figure 4.8 (Cont.): unsupervised results of pageblock dataset continued
Figure 4.9: unsupervised results of wilt dataset
Figure 4.9 (Cont.): unsupervised results of wilt dataset continued
Figure 4.10: unsupervised results of thyroid dataset
Figure 4.10 (Cont.): unsupervised results of thyroid dataset continued
The tables below show the ensemble results based on different base learners.

Table 4.5: unsupervised ensemble results based on knn

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cardio ROC AUC</th>
<th>Cardio AP</th>
<th>PageBlock ROC AUC</th>
<th>PageBlock AP</th>
<th>Wilt ROC AUC</th>
<th>Wilt AP</th>
<th>Thyroid ROC AUC</th>
<th>Thyroid AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>knn</td>
<td>0.89</td>
<td>0.47</td>
<td>0.66</td>
<td>0.22</td>
<td>0.59</td>
<td>0.07</td>
<td>0.81</td>
<td>0.06</td>
</tr>
<tr>
<td>knn_bagging_samples</td>
<td>0.91</td>
<td>0.51</td>
<td>0.66</td>
<td>0.22</td>
<td>0.63</td>
<td>0.07</td>
<td>0.78</td>
<td>0.05</td>
</tr>
<tr>
<td>knn_bagging_features</td>
<td>0.90</td>
<td>0.45</td>
<td>0.64</td>
<td>0.20</td>
<td>0.64</td>
<td>0.07</td>
<td>0.84</td>
<td>0.07</td>
</tr>
<tr>
<td>knn_bagging_k</td>
<td>0.95</td>
<td>0.62</td>
<td>0.66</td>
<td>0.21</td>
<td>0.64</td>
<td>0.07</td>
<td>0.66</td>
<td>0.03</td>
</tr>
<tr>
<td>knn_boosting</td>
<td>0.93</td>
<td>0.57</td>
<td>0.67</td>
<td>0.22</td>
<td>0.66</td>
<td>0.08</td>
<td>0.81</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Table 4.6: unsupervised ensemble results based on knnw

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cardio ROC AUC</th>
<th>Cardio AP</th>
<th>PageBlock ROC AUC</th>
<th>PageBlock AP</th>
<th>Wilt ROC AUC</th>
<th>Wilt AP</th>
<th>Thyroid ROC AUC</th>
<th>Thyroid AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>knnw</td>
<td>0.89</td>
<td>0.39</td>
<td>0.66</td>
<td>0.23</td>
<td>0.76</td>
<td>0.09</td>
<td>0.72</td>
<td>0.04</td>
</tr>
<tr>
<td>knnw_bagging_samples</td>
<td>0.91</td>
<td>0.43</td>
<td>0.66</td>
<td>0.22</td>
<td>0.75</td>
<td>0.09</td>
<td>0.70</td>
<td>0.04</td>
</tr>
<tr>
<td>knnw_bagging_features</td>
<td>0.90</td>
<td>0.39</td>
<td>0.64</td>
<td>0.21</td>
<td>0.68</td>
<td>0.07</td>
<td>0.75</td>
<td>0.04</td>
</tr>
<tr>
<td>knnw_bagging_k</td>
<td>0.95</td>
<td>0.56</td>
<td>0.66</td>
<td>0.22</td>
<td>0.69</td>
<td>0.08</td>
<td>0.70</td>
<td>0.04</td>
</tr>
<tr>
<td>knnw_boosting</td>
<td>0.93</td>
<td>0.58</td>
<td>0.67</td>
<td>0.22</td>
<td>0.70</td>
<td>0.08</td>
<td>0.65</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Table 4.7: unsupervised ensemble results based on odin

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cardio ROC AUC</th>
<th>Cardio AP</th>
<th>PageBlock ROC AUC</th>
<th>PageBlock AP</th>
<th>Wilt ROC AUC</th>
<th>Wilt AP</th>
<th>Thyroid ROC AUC</th>
<th>Thyroid AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>odin</td>
<td>0.83</td>
<td>0.29</td>
<td>0.89</td>
<td>0.18</td>
<td>0.78</td>
<td>0.14</td>
<td>0.72</td>
<td>0.04</td>
</tr>
<tr>
<td>odin_bagging_samples</td>
<td>0.87</td>
<td>0.34</td>
<td>0.88</td>
<td>0.20</td>
<td>0.75</td>
<td>0.12</td>
<td>0.70</td>
<td>0.04</td>
</tr>
<tr>
<td>odin_bagging_features</td>
<td>0.83</td>
<td>0.28</td>
<td>0.90</td>
<td>0.30</td>
<td>0.73</td>
<td>0.09</td>
<td>0.75</td>
<td>0.04</td>
</tr>
<tr>
<td>odin_bagging_k</td>
<td>0.83</td>
<td>0.29</td>
<td>0.89</td>
<td>0.18</td>
<td>0.78</td>
<td>0.14</td>
<td>0.70</td>
<td>0.04</td>
</tr>
<tr>
<td>odin_boosting</td>
<td>0.83</td>
<td>0.37</td>
<td>0.89</td>
<td>0.22</td>
<td>0.71</td>
<td>0.08</td>
<td>0.65</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Table 4.8: unsupervised ensemble results based on lof

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cardio ROC AUC</th>
<th>Cardio AP</th>
<th>PageBlock ROC AUC</th>
<th>PageBlock AP</th>
<th>Wilt ROC AUC</th>
<th>Wilt AP</th>
<th>Thyroid ROC AUC</th>
<th>Thyroid AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>lof</td>
<td>0.91</td>
<td>0.43</td>
<td>0.88</td>
<td>0.25</td>
<td>0.75</td>
<td>0.11</td>
<td>0.64</td>
<td>0.04</td>
</tr>
<tr>
<td>lof_bagging_samples</td>
<td>0.94</td>
<td>0.60</td>
<td>0.68</td>
<td>0.19</td>
<td>0.56</td>
<td>0.06</td>
<td>0.64</td>
<td>0.03</td>
</tr>
<tr>
<td>lof_bagging_features</td>
<td>0.91</td>
<td>0.38</td>
<td>0.90</td>
<td>0.25</td>
<td>0.74</td>
<td>0.09</td>
<td>0.53</td>
<td>0.03</td>
</tr>
<tr>
<td>lof_bagging_k</td>
<td>0.94</td>
<td>0.51</td>
<td>0.90</td>
<td>0.26</td>
<td>0.74</td>
<td>0.09</td>
<td>0.66</td>
<td>0.04</td>
</tr>
<tr>
<td>lof_boosting</td>
<td>0.95</td>
<td>0.65</td>
<td>0.66</td>
<td>0.19</td>
<td>0.67</td>
<td>0.07</td>
<td>0.65</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 4.9: unsupervised ensemble results based on slof

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cardio ROC AUC</th>
<th>Cardio AP</th>
<th>PageBlock ROC AUC</th>
<th>PageBlock AP</th>
<th>Wilt ROC AUC</th>
<th>Wilt AP</th>
<th>Thyroid ROC AUC</th>
<th>Thyroid AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>slof</td>
<td>0.86</td>
<td>0.37</td>
<td>0.87</td>
<td>0.22</td>
<td>0.77</td>
<td>0.11</td>
<td>0.72</td>
<td>0.05</td>
</tr>
<tr>
<td>slof_bagging_samples</td>
<td>0.93</td>
<td>0.55</td>
<td>0.66</td>
<td>0.22</td>
<td>0.57</td>
<td>0.07</td>
<td>0.67</td>
<td>0.04</td>
</tr>
<tr>
<td>slof_bagging_features</td>
<td>0.86</td>
<td>0.33</td>
<td>0.89</td>
<td>0.20</td>
<td>0.73</td>
<td>0.09</td>
<td>0.56</td>
<td>0.03</td>
</tr>
<tr>
<td>slof_bagging_k</td>
<td>0.96</td>
<td>0.70</td>
<td>0.88</td>
<td>0.28</td>
<td>0.69</td>
<td>0.08</td>
<td>0.71</td>
<td>0.05</td>
</tr>
<tr>
<td>slof_boosting</td>
<td>0.96</td>
<td>0.68</td>
<td>0.68</td>
<td>0.22</td>
<td>0.70</td>
<td>0.08</td>
<td>0.74</td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 4.10: unsupervised ensemble results based on inflo

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cardio ROC AUC</th>
<th>Cardio AP</th>
<th>PageBlock ROC AUC</th>
<th>PageBlock AP</th>
<th>Wilt ROC AUC</th>
<th>Wilt AP</th>
<th>Thyroid ROC AUC</th>
<th>Thyroid AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>inflo</td>
<td>0.87</td>
<td>0.40</td>
<td>0.85</td>
<td>0.23</td>
<td>0.78</td>
<td>0.15</td>
<td>0.69</td>
<td>0.04</td>
</tr>
<tr>
<td>inflo_bagging_samples</td>
<td>0.93</td>
<td>0.56</td>
<td>0.64</td>
<td>0.24</td>
<td>0.93</td>
<td>0.12</td>
<td>0.65</td>
<td>0.04</td>
</tr>
<tr>
<td>inflo_bagging_features</td>
<td>0.86</td>
<td>0.38</td>
<td>0.86</td>
<td>0.19</td>
<td>0.77</td>
<td>0.11</td>
<td>0.55</td>
<td>0.03</td>
</tr>
<tr>
<td>inflo_bagging_k</td>
<td>0.96</td>
<td>0.75</td>
<td>0.88</td>
<td>0.25</td>
<td>0.70</td>
<td>0.15</td>
<td>0.67</td>
<td>0.03</td>
</tr>
<tr>
<td>inflo_boosting</td>
<td>0.96</td>
<td>0.71</td>
<td>0.68</td>
<td>0.23</td>
<td>0.71</td>
<td>0.20</td>
<td>0.63</td>
<td>0.05</td>
</tr>
</tbody>
</table>
Conclusion

In this thesis, we demonstrate the effectiveness of ensemble methods for outlier detection. First, we review some algorithms for class imbalance problem in both supervised and unsupervised problems. Next, we present the ensemble frame and some typical methods: bagging and boosting. We also analyze the reason of the enhanced performance of ensemble method in the view of bias-variance tradeoff. Next, the strategies of increasing the diversity of base learners are introduced. Finally, experiment results in various data sets show that bagging and boosting are effective in outlier detection based on various base learners.
Bibliography


