NAÏVE BAYES CLASSIFIER: A MAPREDUCE APPROACH

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

SONGTAO ZHENG

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

October 2014

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Naïve Bayes Classifier: A MapReduce Approach

By

SONGTAO ZHENG

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Saeed Salem

Dr. Yarong Yang

Approved by Department Chair: Dr. Brian Slator

11/14/14	Dr. Brian M. Slator
Date	Signature

ABSTRACT

Machine learning algorithms have the advantage of making use of the powerful Hadoop distributed computing platform and the MapReduce programming model to process data in parallel. Many machine learning algorithms have been investigated to be transformed to the MapReduce paradigm in order to make use of the Hadoop Distributed File System (HDFS). Naïve Bayes classifier is one of the supervised learning classification algorithm that can be programmed in form of MapReduce. In our study, we build a Naïve Bayes MapReduce model and evaluate the classifier on five datasets based on the prediction accuracy. Also, a scalability analysis is conducted to see the speedup of the data processing time with the increasing number of nodes in the cluster. Results show that running the Naïve Bayes MapReduce model across multiple nodes can save considerate amount of time compared with running the model against a single node, without sacrificing the classification accuracy.

ACKNOWLEDGEMENTS

I would like to express my thanks to my adviser, Dr. Simone Ludwig. Her advice, guidance and support helped a lot in my research of Big Data technologies. Also, I sincerely appreciate her time to improve the quality of my paper. I also would like to think the rest of my committee, Dr. Saeed Salem and Dr. Yarong Yang, for their encouragement and insightful comments.

I also thank my parents and grandma for their continuous support during my studying in United States, and thank my girlfriend L. M. for her support during my preparation of my defense.

DEDICATION

This thesis is dedicated to my grandma. For her endless love, support and encouragement through my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1. Big Data

With the prevalence of Web 2.0, Big Data is a heated topic nowadays and is generally believed to be the next big thing in the IT world [1]. Web 2.0 allows the individuals to be connected with each other and share their interpersonal contents online, and enables companies to provide their services online for customers. As a result of the Web 2.0, e-commerce [2] and social networks [3] are starting to build up vast databases of customer activity and wish to generate additional values from it. As well as financial services [4], healthcare [5], telecommunication [6] and many other services, they are beginning to capture more and more data to gain more insights and help them make business decisions. Big Data does not link to any specific quantity of data, but is a general term used to describe the massive amount of digital information a company creates – which takes too much time and costs too much money to be analyzed with traditional computing techniques.

To get a handle on the problems of traditional computing techniques, people have used "three Vs" of volume, velocity and variety to describe the challenges of Big Data [7]. Whereas the volume of data is the biggest challenge of Big Data, it offers us a lot of opportunities. Being able to store and process a vast amount of data raises the possibility of a variety of different activities – disease diagnosis, customer behavior tracking, healthcare services planning, and climate modeling, etc. These tasks usually involve petabytes or even exabytes of data to store and process which goes beyond the capability of traditional database technologies. We need a new system that is capable to overcome the limitations of traditional hardware solutions and scale our Big Data [8]. Velocity is also one of the issues of Big Data. Web 2.0, cloud computing, and the prevalence of mobile devices lead the data to flow into the organizations in an exponential rate that easily dwarfs the traditional computing technologies. Many applications like online video streaming and gaming require the data to be transferred at an exceptionally high velocity, which poses a great pressure to the IT systems [9]. Variety is another characteristic of Big Data, with the digital information many organizations captured becomes increasingly diverse and dense. Voluminous amount of unstructured and semi-structured data that companies create, like photographs, video and audio etc., poses a great challenge with traditional technologies. Big Data in reality is messy and massive data, pre-processing and data cleansing is a necessity before any computing effort can be performed [10].

1.2. Hadoop overview

Due to the challenges brought up by volume, velocity and variety, a new technology is required for Big Data. Apache Hadoop is playing a leading role in the Big Data field currently and it is the first viable platform for Big Data analytics. Hadoop is an open-source software framework for scalable, reliable, distributed computing system that is capable of getting a handle on Big Data's "three Vs" challenges. Originally inspired by Google's MapReduce [11] and Google File System (GFS) [12], what Hadoop does is to use a simple programming model to process large-scale datasets across clusters of machines and distribute the storage. Since the data processing is running on a cluster of machines, it is necessary to deal with node failure that is likely to occur during the course of the processing. Instead of relying on highly expensive servers with high fault tolerance, Hadoop handles node failure itself through its service, which is able to detect the node failure in the cluster and re-distribute the data to other available machines. In addition, Hadoop sets up a scheme to protect it from losing the metadata of the distributed environment. Therefore, Hadoop becomes widely employed by many organizations because of its reliability and scalability to process vast quantities of data with an affordable cost of distributed computing infrastructure [13].

Hadoop consists of two important elements. The first high-performance distributed data processing framework called MapReduce. Hadoop breaks down the datasets into multiple partitions and distribute its storage over the cluster. MapReduce performs data processing on each servers against the blocks of data residing on that machine – which saves a great amount of time due to parallel processing. This emits intermediate summaries which are aggregated and resolved to the final result in a reduce stage. Specifically, the MapReduce paradigm consists of two major steps: map step and reduce step (as shown in Figure 1) – the map step converts the input partition of data into a key/value pair which operates parallel in the cluster, and the reduce task collects the data, performs some computation and resolves them into a single value. When the MapReduce is running on distributed file systems like HDFS, because of HDFS's natural data locality property, the tasks will always operate on the node closest to the residence of the data block. This significantly reduces the I/O cost during the data computation, which allows the parallel data processing to be exceedingly fast [14].

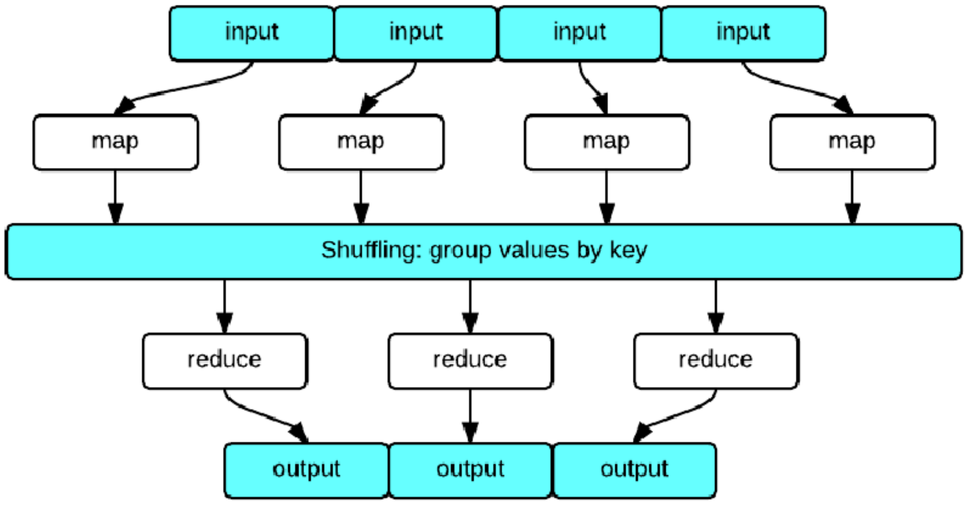

Figure 1. Illustration of the MapReduce framework

The second element of Hadoop is the Hadoop Distributed File System (HDFS), which permits high-bandwidth computation and distributed low-cost storage which is essential for Big Data tasks. Figure 2 shows a basic architecture of HDFS, there is one NameNode and multiple DataNodes, where NameNode manages all of the file system metadata and DataNodes stores the blocks of datasets (represented by b1, b2, b3, etc.). When the datasets are loaded into HDFS, HDFS will distribute the storage of the data across the cluster in a way that will be reliable and can be retrieved faster. A typical HDFS data block size is 64 – 128MB and each partition of data is replicated to multiple nodes. The scale-out architecture gives Hadoop a superb horizontal scalability and significantly increases the availability and fault tolerance of the distributed system [15].

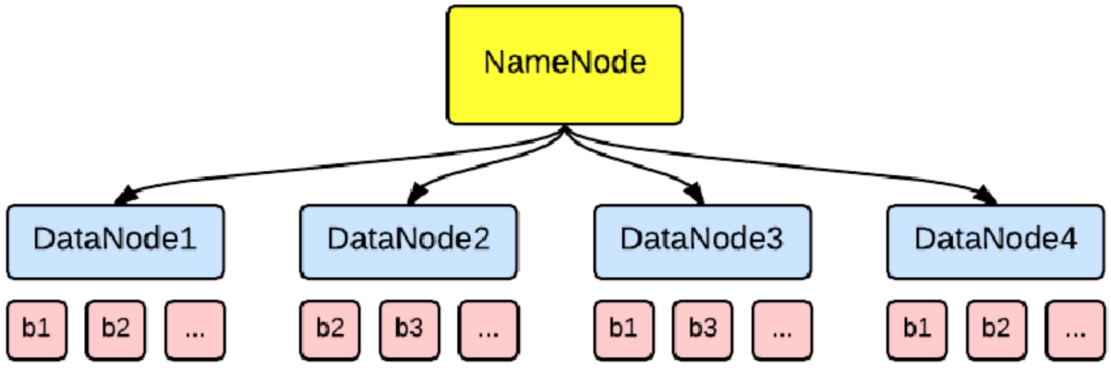

Figure 2. Hadoop Distributed File System (HDFS) architecture

1.3. Word count example

The word count example is a classic example to get started with the Hadoop and MapReduce development. Understanding the word count example is very important for understanding other variations of MapReduce programs. The idea of the word count example is basically to count the number of occurrences of each word of a given input file.

Just like any other MapReduce programs, the word count example also runs in two phases, first the map step on mappers and then the reduce step on reducers. However, before running map and reduce tasks for the word count example, we must first load our input data into the Hadoop Distributed File System (HDFS). Given a large size input file, HDFS will split the input data into several blocks and replicate the blocks to the available nodes in the clusters, which is the key for Hadoop to realize its high availability and fault tolerance.

In the map phase, the content in the given input file will be tokenized and a list of key/value pairs will be formed with key being the each word and value being '1'. For instance, say we have a file with content "hello world bye world", the key-value pairs generated after the map step will be like: <hello, 1>, <world, 1>, <bye, 1>, <world, 1>. In the reduce phase, the intermediate key-value pair from the mapper are sent to the reducers and the pairs with the same key are aggregated and resolved to a single key-value pair. Again, from the above example, the output from the reducers will be like: <hello, 1>, <world, 2>, <bye, 1>. The output gives the information of the words and their occurrences in the given file.

In HDFS, the default block size is 64MB, which means the data will be partitioned into multiple blocks if the size is greater than 64MB. Each block of data will have 3 copies evenly distributed across all the nodes in case of the node failure. Each mapper loads the set of data local to that machine and processes them. Figure 3 gives the illustration of how map and reduce tasks are performed in word count example. For simplicity, we only have two input files and having 4 words in each file. In the map task, we assume the content of each file is stored in separate mappers in HDFS, i.e. "Hello Hadoop" and "Bye Hadoop" are passed to different mapper instances. The mapper instances will process the block data concurrently and emit the intermediate key-value pair. When the mapping phase has completed, there is a shuffle and sort process in which the intermediate key-value pair is exchanged between machines to send all values with the same key to a single reducer. In our example, two <Hadoop, 1> pairs will be placed in the same reducer. In the reduce task, a reducer receives an iterator of key-value pairs and combines them into a single output value. For example, two <Hadoop, 1> pairs will be aggregated as a single key-value pair <Hadoop, 2>.

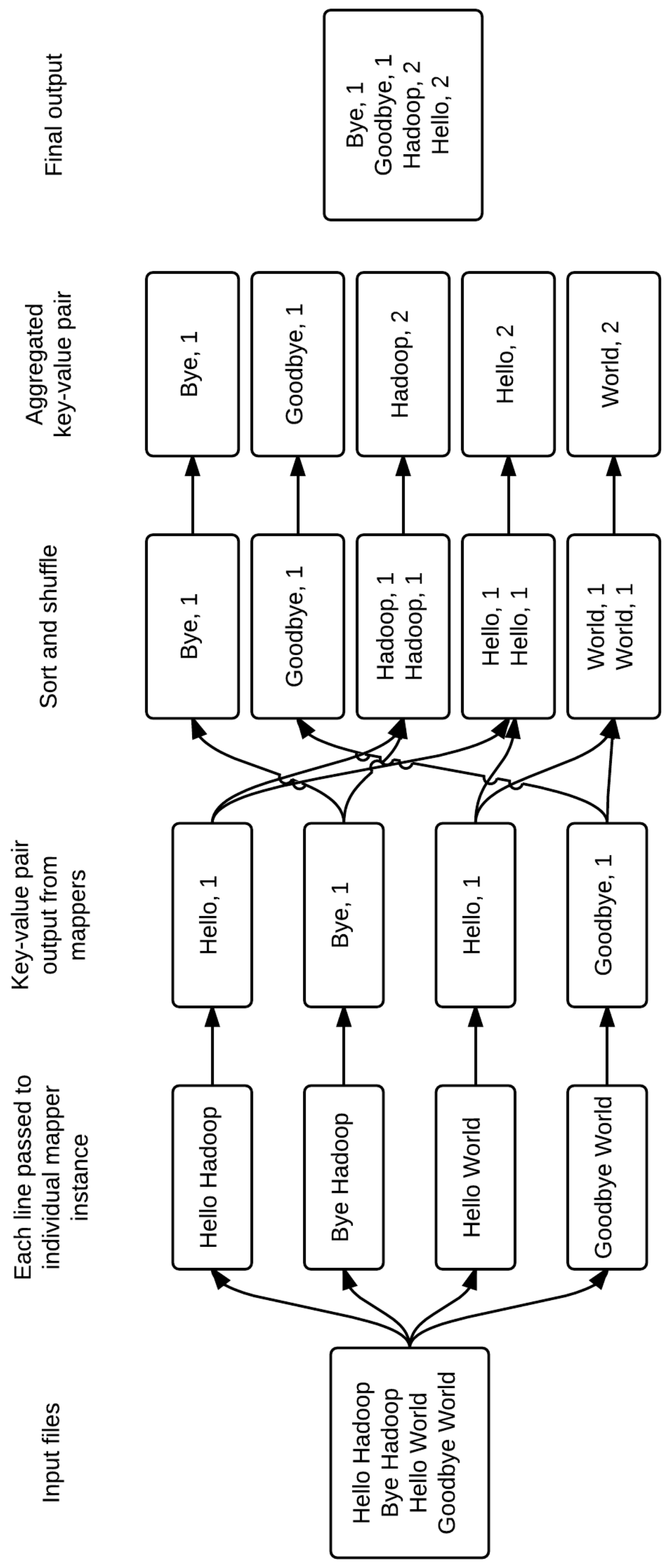

Figure 3. Word count map and reduce flow

2. MACHINE LEARNING AND NAÏVE BAYES

2.1. Large scale machine learning

Machine learning is a technology that has strong ties to statistics and optimization to learn from the existing data to explore hidden valuable information. It has become one of the most popular techniques for knowledge discovery and predictive analytics, especially with the current exponentially growing amount of data from science, business and healthcare, etc. Many applications, like spam filtering [16], advertisement targeting [17], computer vision [18], and bioinformatics [19], etc., have adopted machine learning algorithms to better guide their decisions. Machine learning algorithms can be roughly divided into supervised learning and unsupervised learning. With the domination of distributed computing system in large scale data processing, parallel programming models are developed to speed up machine learning algorithms in multi-core environment [20]. Although some complicated machine learning tasks are difficult to address with the MapReduce paradigm, the MapReduce programming model implemented in the Hadoop platform is the choice to improve the performance for machine learning algorithms [21]. Currently, Mahout - a scalable machine learning library is available to support large data set processing [22].

2.2. Statistical machine learning

2.2.1. Unsupervised learning

Unsupervised learning is used in the problem of learning from a collection of data instances with no training labels. The purpose of unsupervised learning is to discover the under-the-hood cluster pattern of the given collection of data. Clustering is one of the most popular fields of study in unsupervised learning, and it has been widely employed in data mining applications [23]. Clustering techniques have been studied for many years and many learning algorithms have been proposed and improved, such as k-means clustering [24], fuzzy clustering [25], DBSCAN [26], etc.

2.2.2. Supervised learning

Supervised learning considers the problems of estimating a model from data samples with label information. Each input data instance in the sample is associated with corresponding training labels, which is assumed to be a supervised process. A broad family of statistical learning theories have been investigated to achieve risk minimization and generalization maximization in the learning tasks [27]. There are many new machine learning algorithms for applications that are developed based on those statistical theories. Among them, regression based learning is concerned with modeling the relationship between variables that are iteratively refined using a measure of error in the predictions made by the model [28], such as logistic regression, ordinary least squares etc., which have excellent performance in a range of applications, especially for classification tasks. Also, there is kernel based machine learning approach, which is the state-of-art methodology in real-world applications. This approach maps input data into a higher dimensional vector space where some classification or regression problems are easier to model [29]. The example of kernel based algorithms are support vector machine (SVM) and linear discriminant analysis (LDA).

2.3. Naïve Bayes classification algorithm

The Naïve Bayes algorithm is one of the most important supervised machine learning algorithms for classification. This classifier is a simple probabilistic classifier based on applying Bayes' theorem as follows:

$$\mathrm{P(A|B)} = \frac{P(B|A)\,P(A)}{P(B)}$$

Naïve Bayes classification has an assumption that attribute probabilities $\mathrm{P}(x_i|c_j)$ are independent given the class c_j, where x_i is ith attribute of the data instance. This assumption reduces the complexity of the problem to practical and can be solved easily. Despite the simplification of the problem, the Naïve Bayes classifier still gives us a high degree of accuracy.

2.3.1. Formulation

For a data instance d and a class c, based on the Bayes theorem we have:

$$\mathrm{P(c|d)} = \frac{P(d|c)\,P(c)}{P(d)}$$

Where $\mathrm{P(c|d)}$ is the probability of a class given a data instance. $P(d|c)$ is the probability of a data instance given the class, $P(c)$ is the probability of the class, and $P(d)$ is the probability of a document. $\mathrm{P(c|d)}$ is the probability we use to choose the class, specifically, we are looking for a class that maximizes $\mathrm{P(c|d)}$ out of all classes for a given data instance as shown in the following equation:

$$c_{MAP} = \underset{c\in C}{\operatorname{argmax}} P(c|d) = \underset{c\in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)} = \underset{c\in C}{\operatorname{argmax}} P(d|c)P(c)$$

Where c_{MAP} is the class that has maximum a posteriori (MAP), or maximum probability $\mathrm{P(c|d)}$. Notably, the probability of the data instance is a constant, which is dropped from the equation above. We call $P(d|c)$ the likelihood, which is the probability of a given class, and call $P(c)$ the prior, which is the probability of the class. We can represent $P(d|c)$ to be the probability of a vector of attributes conditioning on the class, as follows:

$$c_{MAP} = \underset{c\in C}{\operatorname{argmax}} P(x_1, x_2, \dots, x_n|c)P(c)$$

With the assumption that the attribute probabilities $\mathrm{P}(x_i|c_j)$ are independent given the class c, we have the probability of a set of attributes given the class to be the product of a whole bunch of independent probabilities.

$$P(x_1, x_2, \dots, x_n|c) = P(x_1|c) \cdot P(x_2|c) \cdot P(x_3|c) \cdot \dots \cdot P(x_n|c)$$

Hence the best class of Naïve Bayes classifier will be:

$$c_{NB} = \underset{c\in C}{\operatorname{argmax}} P(c) \prod_{x\in X} P(x|c)$$

2.3.2. Parameter estimation

Upon obtaining the mathematical model of the Naïve Bayes classifier, the next step is to estimate the parameters in the model – the prior of each class and the distributions of attributes. The prior of the class $\mathrm{P(c)}$ can be estimated by simply calculating an estimate for the probability of the class in our training sample:

$$\mathrm{P(c)} = \frac{occurences\ of\ the\ class\ c\ in\ the\ sample}{total\ number\ of\ instances\ in\ the\ sample}$$

With regard to estimation of attribute distributions, there are three models that are widely used in applications – Gaussian Naïve Bayes, Multinomial Naïve Bayes and Bernoulli Naïve Bayes. Gaussian Naïve Bayes is mostly used when dealing with continuous data, while the other two models is well suited for discrete data. For multinomial Naïve Bayes, the probability of ith attribute has value a conditioning on a given class $P(A_i = a|c)$ can be estimated by using the training dataset:

$$P(A_i = a \mid c) = \frac{number\ of\ instances\ having\ i^{th}\ attribute\ A_i\ with\ value\ a}{occurences\ of\ the\ class\ c\ in\ the\ sample}$$

2.3.3. Measurement of Naïve Bayes classifier

The starting point for understanding the measure of the Naïve Bayes algorithm is the following two by two contingency table (Table 1). There are essentially four states for any particular piece of data we evaluate. On one axis we choose whether this piece of data correctly belongs to a class or whether it does not correctly belong to a class, and thus this axis we describe as truth. Now, we build a Naïve Bayes classifier that tries to detect the truth and the classifier will tell us the data belongs to which class. Therefore, there are four possibilities that occur – true positive (tp), true negative (tn), false positive (fp) and false negative (fn). For example, if the truth of the data instance is correct and we select it, then it is a true positive. Another possibility is that our classifier does not say it it correct, then it is a false negative. On the other hand, it is possible that the instance is not correct in which case there are two possibilities – our classifier mistakenly classifies the instance to the "correct" category, which is called false positive, or our classifier correctly places the instance into the "not correct" category, which is called true negative.

Table 1. The 2-by-2 contingency table

	correct	not correct
selected	tp	fp
not selected	fn	tn

1) Accuracy

Accuracy is the first reasonable measure to look at. Accuracy equals the true positives, plus the true negatives over all four classes (true positives, true negatives, false positives and false negatives). In many applications, accuracy is a useful measure for the Naïve Bayes classifier. But there is a particular scenario when dealing with things that are uncommon, in which the accuracy is not a useful measure. For example, 99.99% of the data are from category A, while only 0.01% is from the counterpart category B. So, we will have 99.99% of accuracy even when our classifier assigned all data into category B, which is apparently undesired. For this situation, precision and recall are used as the measurement of our classifier.

$$\text{Accuracy} = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ positives + false\ negatives}$$

2) Recall

Recall is the percentage of correct items that are selected. By applying recall as a measurement, the above-mentioned situation for accuracy can be resolved. Simply assigning all items to category A will give zero recall, which turns out to be an undesired classifier. In various applications, such as for things like legal applications, where you want to find all of the appropriate evidence, such as in discovery procedures. What you really want to do is having a classifier that has high recall, that finds as much of the relevant items as possible. In other words, you do not want relevant evidences left unselected.

$$\text{Recall} = \frac{true\ positives}{true\ positives + false\ negative}$$

3) Precision

Precision is the percentage of selected items that are correct. Precision can also be a resolution for the condition when dealing with uncommon things. In some contexts, you might be more interested in precision over recall. For example, you want to show customers some merchandises that are good (correct), and do not care that only 1/10 or 1/20 of the things do satisfy their query.

$$\text{Precision} = \frac{true\ positives}{true\ positives + false\ positives}$$

We see the tradeoff that balances between recall and precision, because inevitably if you increase your recall then you are going to make some mistakes hence the precision goes down. The more you try and boost recall, the more your precision is starting to drop. So people are trading off precision and recall and the tradeoff is played out differently in different applications. If it is important to find the correct items returned then you should choose to have a high precision, while you will need a high recall classifier if you try to find all correct things. But sometimes people want to determine which classifier is better and need a way to compare, and the F measure is proposed to combine precision and recall.

4) F measure

F measure is a combined measure that assesses the precision and recall tradeoff and is basically a weighted harmonic mean of precision (P) and recall (R):

$$\text{F} = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

Where α is the coefficient that weights precision or recall, people usually use balanced F1 measure as the measurement:

$$\text{F1} = \frac{2PR}{P+R}$$

2.3.4. Validation of Naïve Bayes classifier

If we use the same data set for both the training and testing phase of the algorithm, the outcome will be overly optimistic [30]. Hence in order to obtain a better estimate of the model parameters and simulate real-world situation, the testing sample is not included in the training and acts as "unknown" data. After the training phase is completed, then we apply the model against the testing sample to see the outcome accuracy of the model.

1) Holdout validation

Holdout method is the simplest validation method, in this method the original data set is partitioned into a training and testing set. The training set is solely used for model parameter estimation, and the testing set is used for the validation of the model prediction accuracy [31]. This method works when there are no dependencies in the chosen training sample, however, this is not always guaranteed. It is very likely that the instances in the training sample are correlated, which causes a bias in the model estimate.

2) K-fold cross validation

In order to improve the estimate of holdout validation, k-fold cross validation is introduced. In this method, the data set is divided into K subsets, in each of K runs, 1 subset is used for testing and the remaining $\mathrm{K}-1$ subsets are used for training. The error rate of the model is the average of all K error rates. After K runs, we choose the model which has the lowest error rate and apply it against the "unknown" data to see the outcome accuracy. Although k-fold cross validation cost K times computation compared with holdout validation, but it opts out the possible bias in the model estimation.

3. IMPLEMENTATION

3.1. Overview

In this section, we describe the implementation and development details of building the Naïve Bayes classifier using the MapReduce paradigm. The implementation contains five steps: data preparation, data preprocessing, data processing using MapReduce, data classification using Naïve Bayes algorithm, and classifier evaluation. We downloaded 5 datasets (*adult*, *car-evaluation*, *contraceptive method choice*, *mushroom*, and *nursery*) from the University of California Irvine (UCI) Machine Learning Repository [32], all of the datasets are multivariate of categorical or integer variables and default for classification tasks. In the data preprocessing stage, we augmented the original datasets to gigabyte level so to better leverage the powerful Hadoop computing capability. Also, the dataset is divided into two parts: one for model training and the other are treated as "unknown" data for model testing. We use cross validation to find the estimated model with the highest accuracy. If there are continuous attributes in our dataset, a preprocessing MapReduce job is performed to discretize the continuous variables. After preprocessing is done, we load the preprocessed data into HDFS, and perform map and reduce tasks against the data. In this step, we are able to obtain the number of occurrences of an attribute with a specific value given a certain class. With the result from the MapReduce task, we can compute the prior, likelihood, and hence posterior for each class, and then assign the instance to a class. Finally, we count the number of instance correctly and wrongly classified to get an overall accuracy of the estimated model. The overview of the implementation diagram is shown in Figure 4.

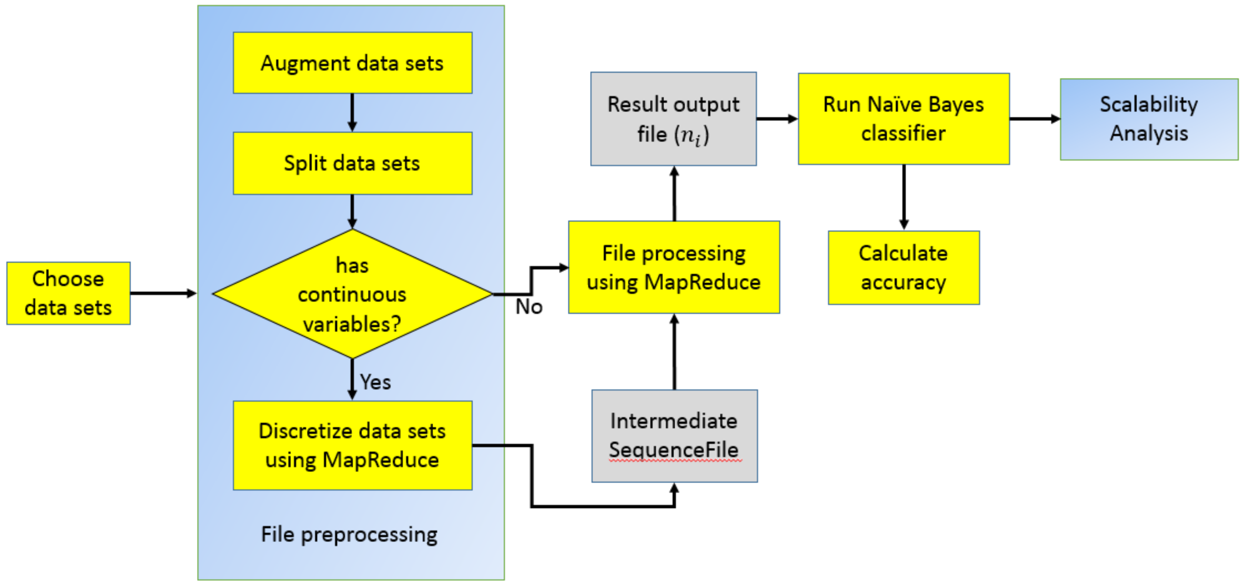
Choose data sets
Augment data sets
Split data sets
has continuous variables?
Yes
No
Discretize data sets using MapReduce
File preprocessing
Intermediate SequenceFile
File processing using MapReduce
Result output file (n_i)
Run Naïve Bayes classifier
Calculate accuracy
Scalability Analysis

Figure 4. Implementation overview

3.2. Implementation choice

Hadoop is written in Java, although Hadoop streaming supports many languages to enjoy the advantages offered by Hadoop platform, MapReduce is still most commonly written in Java [33]. The Hadoop version I used is cdh4.2.0 – a Cloudera distribution of Apache Hadoop. The project is built and packaged using Apache Maven. In our implementation, we construct explicit mapper and reducer functions in Java. My choice of development environment is Eclipse.

3.3. File preprocessing

3.3.1. Data augmentation

Hadoop is designed for large scale data processing while suffering a performance penalty when dealing with small datasets [34, 35]. If you have data measured in terabyte or even petabyte, the superior scalability of Hadoop will save you a considerable amount of money and time. However, if you only have megabytes of data, then Hadoop is not the choice as its performance will be significantly lower and is no competitor for Excel or other SQL tools. Hadoop Wiki suggests that since the map/reduce setup takes for a while, so ideally we should at least allow the map task to execute for a minute [36]. The five datasets downloaded from UCI Machine Learning Repository only have their size at a few megabytes level, which is not suitable for Hadoop experiment. Hence, I augmented the datasets by concatenating the copy of data to a much greater size so that Hadoop can distinguish the environment warm-up time and the actual data processing time. The size of the augmented data is still not ideal for Hadoop to show its strong scaling capability, which usually goes beyond terabytes or even exabytes, however, this is sufficient for our experiments.

3.3.2. Data split and cross validation

In order to test the accuracy of our Naïve Bayes classifier, we randomly take 70% of instances for model training and the remaining 30% for model testing. The testing data set is treated as "unknown" data, which is never involved in the training process. This is considered to be a resemble of the real-world and will not get overoptimistic estimates. The training dataset is divided into 6 folds and cross validation is performed to eliminate the bias within the dataset. There are in total 6 runs for cross validation and in each run we in turn pick 1 of 6 folds as the testing set and the rest for training. Finally we choose the estimated model with lowest error rate. The selected model will be applied to the "unknown" data for classification. We calculate the accuracy of the classifier based on the number of instances correctly and wrongly estimated.

3.3.3. Data discretization

There are two of five datasets – *adult* and *contraceptive method choice* - that have mixed categorical and continuous variables, and therefore, we have to discretize the continuous variables in the datasets so as to apply the Naïve Bayes classifier. Instead of using the Gaussian distribution assumption to estimate mean and variance of continuous variables, we use binning to categorize continuous feature attributes, to obtain a new set of Bernoulli-distributed attributes [37]. The number of bins can be customized, but usually a more finely spaced bin can reduce the discretization error with an increasing cost of computation [38]. In order to correctly categorize continuous variables, it is required to obtain the range of the variable. We dump the data into HDFS and perform an additional

MapReduce job to find the maximum and minimum value of each continuous variable. This step saves us additional time compared with the traditional approach because the computation can be done in parallel across the cluster. Below is the snippet of MapReduce source code in the preprocessing step:

The mapper class source code is shown in Figure 5.

```
public class PreprocessMapper extends Mapper<LongWritable, Text, Text, 
DoubleWritable> {
    protected void map(LongWritable key, Text value, Context context) throws 
IOException, InterruptedException
    {...}
}
```

Figure 5. PreprocessMapper.java

Input key: the offset of the beginning of the line from the beginning of the file [39]

Input value: the string content of the line

Output key: attribute name (i.e. age)

Output value: attribute value (i.e. 29)

Where only continuous variables and their values are sent to the reducers.

The reducer class source code is shown in Figure 6.

```
public class PreprocessReducer extends Reducer<Text, DoubleWritable, Text, 
DoubleWritable>
{
    protected void reduce(Text key, Iterable<DoubleWritable> values, Context 
context) throws IOException, InterruptedException
    {...}
}
```

Figure 6. PreprocessReducer.java

Input key: attribute name

Input value: attribute value

Output key: attribute name_min or attribute name_max (i.e. age_min, age_max)

Output value: minimum or maximum value of the attribute

Notably, the output key/value of the mapper should be identical to the input key/value of the reducer. The reduce function takes a list of value output from the map function and resolves to a single maximum and minimum value of a continuous variable.

We store the output from the preprocessing step as a SequenceFile, a flat file consisting of binary key/value pairs, which is also internally used for temporary map output [40]. Using SequenceFile as the intermediate output format increases the performance when there are multiple MapReduce jobs run in consecutive.

The remaining three datasets (*car-evaluation*, *mushroom* and *nursery*) do not have continuous feature attributes, hence, there is no need to run data discretization step on those three datasets.

3.4. File processing and parameter estimation

File processing is the key step for our Naïve Bayes classifier implementation. We build a MapReduce model to estimate the parameters in the model – prior and likelihood of a given class. The model is able to handle large scale Naïve Bayes classification tasks due to the distributed nature of the MapReduce paradigm.

3.4.1. Naïve Bayes MapReduce model

Naïve Bayes is one of the machine learning algorithms that can be applied against the MapReduce paradigm [41]. For simplicity, suppose there are only two classes c_1 and c_2 for classification and each data instance has M attributes (mostly with discrete value). In order to estimate the parameters in the Naïve Bayes model, i.e. $\mathrm{P}(c_j)$ and $P(A_i = a|c_j)$, where $\mathrm{P}(c_j)$ is the prior of class c_j ($\mathrm{j} = 1$ or 2), and $P(A_i = a|c_j)$ is the likelihood of the ith attribute, A_i has value a ($\mathrm{i} = 1, 2, \dots, \mathrm{M}$) conditioning on class c_j, we need to get the total number of instances in the sample, namely N, occurrences of class c_j in the sample, namely N_j, and number of instances having ith attribute A_i with value a in the sample namely n_i. By iterating through the sample, we can obtain N and N_j.

For the next step we use the MapReduce model to obtain the value of n_i. In the map task it takes a line as input and converts the content of the line into a key/value pair, where the key is a combination of the class, attribute and its attribute value, namely a unique string with a form like $A_i\_a_i\_c_j$, the value is 1. The source code snippet of the map function is shown in Figure 7.

```
public class ProcessMapper extends Mapper<LongWritable, Text, Text, 
IntWritable>
{
    protected transient HashMap<String, Double> _map;
    protected void setup(Context context) throws IOException, 
InterruptedException
    {...}
    protected void map(LongWritable key, Text value, Context context) throws 
IOException, InterruptedException
    {...}
    private void createContKey(String attribute, String value, String 
category, int bin, Context context) throws IOException, InterruptedException
    {...}
    private void createDiscKey(String attributes, String value, String 
category, Context context) throws IOException, InterruptedException
    {...}
}
```

Figure 7. ProcessMapper.java

Input key: the offset of the beginning of the line from the beginning of the file

Input value: the string content of the line

Output key: attribute name_attribute value_class (i.e. age_0_<=50K, education_Masters_>50K)

Output value: 1

In the reduce task, the values of the same key is added up and a single key/value pair is emitted, where the key is the unique string combination and the value is the count of the occurrences of such string combination. The reducer class source code is shown in Figure 8.

```
public class ProcessReducer extends Reducer<Text, IntWritable, Text, 
IntWritable>
{
    protected void reduce(Text key, Iterable<IntWritable> values, Context 
context) throws IOException, InterruptedException
    {...}
}
```

Figure 8. ProcessReducer.java

Input key: attribute name_attribute value_class (i.e. age_0_<=50K, education_Masters_>50K)

Input value: 1

Output key: attribute name_attribute value_class (i.e. age_0_<=50K, education_Masters_>50K)

Output value: number of instances having ith attribute A_i with value a in the sample n_i

The reduce function takes a list of 1's from the map function and resolves to a single count.

3.4.2. Parameter estimation and classification

With the MapReduce approach, we are able to process the file in multiple nodes in parallel and significantly expedite the performance compared with running on single node. From the output of the MapReduce run, we can easily obtain the prior and likelihood for each class by:

$$\mathrm{P}(c_j) = \frac{N_j}{N}$$

$$\mathrm{P}(A_i = \mathrm{a} \mid c_j) = \frac{n_i}{N_j}$$

For every instance d, according to Bayes' theorem we have the posterior for instance d:

$$\mathrm{P}(c_j|\mathrm{d}) \propto \mathrm{P}(c_j) * \mathrm{P}(A_1 = a_1 \mid c_j) * \mathrm{P}(A_2 = a_2 \mid c_j) * \dots * \mathrm{P}(A_M = a_M \mid c_j)$$

If $\mathrm{P}(c_1|\mathrm{d})$ is greater than $\mathrm{P}(c_2|\mathrm{d})$, we classify data instance d to class c_1, otherwise we classify it to c_2. Sometimes it is necessary to use log likelihood instead of likelihood to avoid floating point overflow, then we have:

$$\mathrm{P}(c_j|\mathrm{d}) \propto \log \mathrm{P}(c_j) + \log \mathrm{P}(A_1 = a_1 \mid c_j) + \log \mathrm{P}(A_2 = a_2 \mid c_j) + \cdots + \log \mathrm{P}(A_M = a_M \mid c_j)$$

Notably, it is possible that an attribute value does not occur in every class, which will lead to a zero likelihood estimate for that class. To get a handle on this zero-frequency issue, we added a value 1 to the count of every attribute value-class combination. Also, as discussed in Section 4.3.3, if we have continuous feature attributes that exist, a preprocessing job is required to categorize the continuous variable before we estimate the parameters using the MapReduce model.

3.4.3. Model evaluation

In the cross validation step, we apply the Naïve Bayes MapReduce model developed in our study for each of the 6 runs and select the model with the highest accuracy. The chosen model will be used to classify the "unknown" data. Finally, we count the number of instances correctly and wrongly classified, and hence, obtain the model accuracy. If the dataset has only two classes, we can also calculate the recall, precision, and F measure for our classifier.

3.5. Scalability analysis

Horizontal scalability is necessary for large-scale data processing. By adding more nodes to the system, the computation can be distributed across the nodes and run in parallel [42]. This not only significantly reduces the hardware cost, but also processing time. In this section, we will conduct a scalability analysis to discover how Hadoop horizontal scalability improves the performance of the system, and the relationship between the processing time and the number of nodes used in the system. Notably, map and reduce tasks take time to warm up, so it is recommended that the task takes at least one minute to run. The implementation of the Naïve Bayes algorithms have two MapReduce jobs – preprocess file, and process file – here we only study the second MapReduce job. Different number of nodes are used to process the data, i.e. single node and 2, 4, 6, 8, 10, 12, 14 nodes are used for each run to split the data to be sent to the mappers (shown in Table 2). Number of reduce tasks actually spawned will be less than the number of mappers but determined by Hadoop internally. The running time is recorded by placing a timestamp at the beginning and end of the method. Table 2 shows the number of mappers and reducers used for each run. Since node failure could happen and there are many variables that could impact the running time, for each node setting we perform 10 runs and take the average as the running time reported.

Table 2. Number of nodes used in each run

Run	1	2	3	4	5	6	7	8
# Mappers	1	2	4	6	8	10	12	14

In our implementation the total number of nodes in use is configured as a command line variable, as well as the input file path and output directory. The example command to run this algorithm is as follows:

```
hadoop jar job/nb-classifier-1.0.jar NaiveBayesJob -inputfile ~/naive-bayes/input/adult-training-1.data -outputdir ~/naive-bayes/output -numnodes 8
```

Where job/naïve-bayes-1.0.jar is the job jar file containing the binary code of the implementation

NaiveBayesJob is the class containing main method

-inputfile is the input file path

-outputdir is the output directory

-numnodes is the number of nodes in use for each run

Notably all nodes in use will be used as mappers, usually half of the nodes or more will be used as reducers during the MapReduce run.

4. EXPERIMENTS AND RESULTS

This section describes two experiments. The first is experiment is the evaluation of the Naïve Bayes classifier build on top of the MapReduce model. Five datasets downloaded from the UCI Machine Learning Repository are used in our experiments (Table 3). The second experiment is the scalability analysis to see the relationship between the number of nodes used in the system and the processing time.

Table 3. Five experiment datasets

Name	Data Types	Attribute Types	# Instances	# Attributes
Adult	Multivariate	Categorical, Integer	48842	14
Car evaluation	Multivariate	Categorical	1728	6
Contraceptive Method Choice	Multivariate	Categorical, Integer	1473	9
Mushroom	Multivariate	Categorical	8124	22
Nursery	Multivariate	Categorical	12960	8

4.1. Naïve Bayes MapReduce model evaluation

4.1.1. Experiment 1 – adult dataset

The adult dataset has 48842 instances and 14 attributes, including 6 continuous variables and 8 categorical variables. There are two classes in this dataset, the individual income is greater than 50K (>50K) or no greater than 50K (<=50K). Table 4 is a snapshot of adult data attributes. All continuous attributes are discretized into 4 bins so that the Naïve Bayes algorithm can be employed.

The original data is divided into training and testing set, where the training set has 70% of the data and testing set has 30%. 6-fold cross validation is used for parameter estimation. In the first run, we use the first fold as testing set, for the second run we use the second fold as testing set, and so on. The result accuracy rate for each run is shown in Table 5.

Table 4. Attribute and its type of adult data

Attribute	Type
age	continuous
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
fnlwgt	continuous
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
education-num	continuous
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transportation-moving, Priv-house-serv, Protective-serv, Armed-Forces
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	Female, male
capital-gain	continuous
capital-loss	continuous
hours-per-week	continuous
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands

Table 5. Accuracy of 6 runs of cross validation for adult data

Run	1	2	3	4	5	6
Accuracy rate	0.762563	0.763433	0.763625	0.762413	0.762104	0.76257
Error rate	0.237437	0.236567	**0.236375**	0.237587	0.237896	0.23743

From Table 5, we can see that the accuracy rate of each run is very close, we can say that there is no bias for each training set. As the error rate of the third run is slightly lower than the rest of the runs, we use the estimated model of this run to predict the classification of the “unknown” data. The two-by-two contingency table of the Naïve Bayes classification result is shown in Table 6.

Table 6. The 2-by-2 contingency table of adult data

	Actual annual income > 50K	Actual annual income <= 50K
Classified to “>50k”	tp=612606	fp=599740
Classified to “<=50K”	fn=94603	tn=1623541

Here we define “Actual annual income > 50K” to be true, while “Actual annual income <= 50K” to be false. With the contingency table, we can calculate several measurement to evaluate the Naïve Bayes classifier.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} = 0.763062$$

$$Recall = \frac{tp}{tp + fn} = 0.86623$$

$$Precision = \frac{tp}{tp + fp} = 0.505306$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = 0.638279$$

For the adult dataset, we are equally concerned about whether we can correctly classify each individual to the actual annual income class, no matter >50K or <=50K. So here accuracy is our interest of measurement to evaluate the model, 76% of accuracy is an acceptable level to predict the income class of an individual.

4.1.2. Experiment 2 – car evaluation dataset

The car evaluation dataset has 1728 instances and 6 attributes and all attributes are categorical variables, hence there is no need to discretize the data. There are four classes in this dataset – unacc (unacceptable), acc (acceptable), good and vgood (very good). Table 7 is a snapshot of car evaluation data attributes.

Table 7. Attribute and its type of car evaluation data

Attribute	Type
buying	v-high, high, med, low
maint	v-high, high, med, low
doors	2, 3, 4, 5-more
persons	2, 4, more
lug_boot	small, med, big
safety	low, med, high

From Table 8, we can see the error rate of the second run is slightly lower than the rest of the runs, we use the estimated model of this run to predict the classification of the "unknown" data. The four-by-four confusion table of the Naïve Bayes classification result is shown in Table 9. The accuracy of the Naïve Bayes classifier is 0.8739.

Table 8. Accuracy of 6 runs of cross validation for adult data

Run	1	2	3	4	5	6
Accuracy rate	0.872490079	0.87409623	0.872574405	0.872980159	0.872780754	0.873262897
Error rate	0.127509921	**0.12590377**	0.127425595	0.127019841	0.127219246	0.126737103

Table 9. The 4-by-4 confusion table of car evaluation data

	unacc	acc	good	vgood
Classified to "unacc"	1742470	127760	0	0
Classified to "acc"	70432	433176	68560	39016
Classified to "good"	3010	15044	31261	0
Classified to "vgood"	0	0	3021	58250

$$
\begin{aligned}
&Accuracy \\
&= \frac{1742470 + 433176 + 31261 + 58250}{1742470 + 433176 + 31261 + 58250 + 127760 + 70432 + 68560 + 39016 + 3010 + 31261 + 3021} \\
&= 0.8739
\end{aligned}
$$

We can see a very promising result accuracy of the classifier that 87.39% of the unknown cars are correctly classified to its actual acceptance level. Also, from the 4-by-4 confusion table, we can see that only a very small portion of unacceptable cars are accepted and no good or very good cars are treated as unacceptable cars.

4.1.3. Experiment 3 – contraceptive method choice dataset

The contraceptive method choice dataset has 1473 instances and 9 attributes including 2 continuous variables and 7 categorical variables. There are three classes in this dataset, No-use (1), Long-term (2) and Short-term (3). Table 10 is a snapshot of contraceptive method choice data attributes. Attribute "Wife's age" is discretized into 5 categories, while attribute "Number of children ever born" is discretized into 8 categories.

From Table 11, we can see the error rate of the second run is slightly lower than the rest of the runs, we use the estimated model of this run to predict the classification of the "unknown" data. The four-by-four confusion table of the Naïve Bayes classification result is shown in Table 12. The accuracy of the Naïve Bayes classifier is 0.5182.

Table 10. Attribute and its type of contraceptive method choice data

Attribute	Type
Wife's age	continuous
Wife's education	1=low, 2, 3, 4-high
Husband's education	1=low, 2, 3, 4-high
Number of children ever born	continuous
Wife's religion	0=Non-Islam, 1=Islam
Wife's now working?	0=Yes, 1=No
Husband's occupation	1, 2, 3, 4
Standard-of-living index	1=low, 2, 3, 4-high
Media exposure	0=Good, 1=Not good

Table 11. Accuracy of 6 runs of cross validation for cmc data

Run	1	2	3	4	5	6
Accuracy rate	0.51841	0.481698	0.481374	0.48316	0.481007	0.48087
Error rate	**0.48159**	0.518302	0.518626	0.51684	0.518993	0.51913

Table 12. The 4-by-4 confusion table of contraceptive method choice data

	No-use	Long-term	Short-term
Classified to "No-use"	1224569	303302	592985
Classified to "Long-term"	378241	551109	426720
Classified to "Short-term"	284863	143166	514045

$$Accuracy = \frac{1224569 + 551109 + 514045}{1224569 + 551109 + 514045 + 303302 + 592985 + 378241 + 426720 + 284863 + 143166} = 0.5182$$

For this dataset, the Naïve Bayes classifier only classifies 51.82% of the unknown data correctly, which is not a very promising result. There could be a couple of reasons that the Naïve Bayes classifier is the best very fit for the dataset – the number of instances is not sufficient, the attributes are confounded or dependent which degrades its usefulness. We can use other classification algorithms to classify this dataset, i.e. linear or logistic regression, support vector machine (SVM) etc.

4.1.4. Experiment 4 – mushroom dataset

The mushroom dataset has 8124 instances and 22 attributes and all attributes are categorical variables, hence, there is no need to discretize the data. There are two classes in this dataset – edible mushroom and poisonous mushroom. Table 13 is a snapshot of mushroom data attributes.

Table 13. Attribute and its type of mushroom data

Attribute	Type
cap-shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
cap-surface	fibrous=f, grooves=g, scaly=y, smooth=s
cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
bruises?	bruises=t, no=f
odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
gill-attachment	attached=a, descending=d, free=f, notched=n
gill-spacing	close=c, crowded=w, distant=d
gill-size	broad=b, narrow=n
gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
stalk-shape	enlarging=e, tapering=t
stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
stalk-surface-above-ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk-surface-below-ring	fibrous=f, scaly=y, silky=k, smooth=s
stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
veil-type	partial=p, universal=u
veil-color	brown=n, orange=o, white=w, yellow=y
ring-number	none=n, one=o, two=t
ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

From Table 14, we can see the error rate of the third run is slightly lower than the rest of the runs, we use the estimated model of this run to predict the classification of the "unknown" data. The two-by-two contingency table of the Naïve Bayes classification result is shown in Table 15. The accuracy of the Naïve Bayes classifier is 0.9929.

Table 14. Accuracy of 6 runs of cross validation for adult data

Run	1	2	3	4	5	6
Accuracy rate	0.992893	0.992998	0.993112	0.993093	0.992946	0.993043
Error rate	0.007107	0.007002	**0.006888**	0.006907	0.007054	0.006957

Table 15. The 2-by-2 contingency table of mushroom data

	Actual poisonous mushroom	Actual edible mushroom
Classified to "poisonous"	tp=1396965	fp=7327
Classified to "edible"	fn=13400	tn=1506948

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} = 0.9929$$

$$Recall = \frac{tp}{tp + fn} = 0.9905$$

$$Precision = \frac{tp}{tp + fp} = 0.9948$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = 0.9926$$

Nearly all the unknown instances are correctly classified and our Naïve Bayes classifier works perfect for the mushroom dataset. Four measurements – accuracy, recall, precision and F measure are all above 99%. This is probably due to the fact that the dataset has only two classes for classification and the number of each class in the sample is very close, also we have 22 feature attributes for the classification task.

4.1.5. Experiment 5 – nursery dataset

The nursery dataset has 12960 instances and 8 attributes and all attributes are categorical variables, hence there is no need to discretize the data. There are five classes in this dataset – not_recom, recommend, very_recom, priority and spec_prior. Table 16 is a snapshot of nursery data attributes.

From Table 17, we can see the error rate of the third run is slightly lower than the rest of the runs, we use the estimated model of this run to predict the classification of the "unknown" data. The five-by-five confusion table of the Naïve Bayes classification result is shown in Table 18. The accuracy of the Naïve Bayes classifier is 0.9016.

Table 16. Attribute and its type of nursery data

Attribute	Type
parents	usual, pretentious, great_pret
has_nurs	proper, less_proper, improper, critical, very_crit
form	complete, completed, incomplete, foster
children	1, 2, 3, more
housing	convenient, less_conv, critical
finance	convenient, inconv
social	non-prob, slightly_prob, problematic
health	recommended, priority, not_recom

Table 17. Accuracy of 6 runs of cross validation for adult data

Run	1	2	3	4	5	6
Accuracy rate	0.901551834	0.90082523	0.901657654	0.901176504	0.901504711	0.901261656
Error rate	0.098448166	0.09917477	**0.098342346**	0.098823496	0.098495289	0.098738344

Table 18. The 5-by-5 confusion table of nursery data

	not_recom	recommend	very_recom	priority	spec_prior
Classified to "not_recom"	1036881	0	0	0	0
Classified to "recommend"	0	0	0	0	0
Classified to "very_recom"	0	0	0	0	0
Classified to "priority"	0	460	78353	934263	137911
Classified to "spec_prior"	0	0	0	89381	833144

$$Accuracy = \frac{1036881 + 934263 + 833144}{1036881 + 934263 + 833144 + 460 + 78353 + 137911 + 89381} = 0.9016$$

The accuracy our classifier is at a level of 90%, we could say the model is more than enough to predict the unknown nursery data. However, from the confusion table we can observe that there is no "recommend" or "very recommend" data that is correctly classified, in other words, all recommended and very recommended nurses are classified into other groups. This is probably due to that there is only 0.015% recommended nurses, and 2.531% very recommended nurses in our sample dataset. We can use other classifiers like support vector machine (SVM), artificial neural networks (ANN), decision tree model, random forest model, etc. for prediction if one or more classes are only a very small portion of the entire dataset.

4.2. Scalability analysis

The scalability analysis of MapReduce paradigm is performed against the second MapReduce job run – file processing. The MapReduce jobs are executed on Hadoop cluster hosted in Department of Computer Science at North Dakota State University. The cluster has a capacity of 14 virtual machines in total, of which all can be run as a mapper or reducer. As discussed in Section 4.6, we have 8 number of nodes setting to perform the analysis, and in each setting we run the MapReduce job 10 times because many causes which beyond our control could impact the job running time. The running time, recorded in seconds, for each nodes setting is shown in Table 19. The relationship between the processing time and number of nodes used in the system is displayed in Figure 9.

Table 19. Running time of each MapReduce run (adult dataset)

# of nodes	1	2	4	6	8	10	12	14
Run 1	580.138	247.563	144.346	180.343	173.264	166.43	170.419	109.236
Run 2	628.37	212.438	143.553	159.669	156.263	160.416	135.408	96.39
Run 3	461.847	296.547	207.753	193.795	125.923	150.682	147.507	121.023
Run 4	414.454	356.884	178.595	141.294	167.455	101.43	104.783	101.488
Run 5	489.227	229.743	239.913	185.745	139.22	142.497	86.317	105.222
Run 6	417.212	360.798	158.418	152.442	125.362	124.304	113.547	127.382
Run 7	423.052	354.824	207.46	123.527	132.687	108.234	105.148	122.596
Run 8	513.744	215.532	123.767	174.659	171.454	105.693	143.478	119.276
Run 9	449.821	313.728	189.82	129.285	111.402	140.361	144.549	125.232
Run 10	475.865	309.569	214.681	165.445	95.435	123.343	114.668	110.884
Average	485.373	289.7626	180.8306	160.6204	139.8465	132.339	126.5824	113.8729
Std. dev.	71.27059	59.31965	37.46947	23.89855	26.6339	23.20544	25.63301	10.70952

From Figure 9 we can see that the job running time drops significantly from single node execution to 4-node processing – in single node run, the node will be used as both mapper and reducer, the data flow and I/O will be significantly higher and it could possibly suffers from network interruption or outages. With the increasing number of nodes, the impact of node failure can be mitigated and jobs can be run in parallel for each block of the data. A 4-node cluster only takes 180 seconds to complete the job as compare to 483 seconds for single node processing. More nodes can be added to the cluster leveraging the horizontal scalability of Hadoop, the job performance can be improved even more with the addition of new nodes. However, the improvement is not as significant as the beginning, a 14-node cluster is only 67 seconds faster than a 4-node cluster. One reason is that the size of our data is not very large, a 4-node cluster is sufficient for our needs of scalability. From the experiment, we can observe the strong scalability of Hadoop for large-scale data processing.

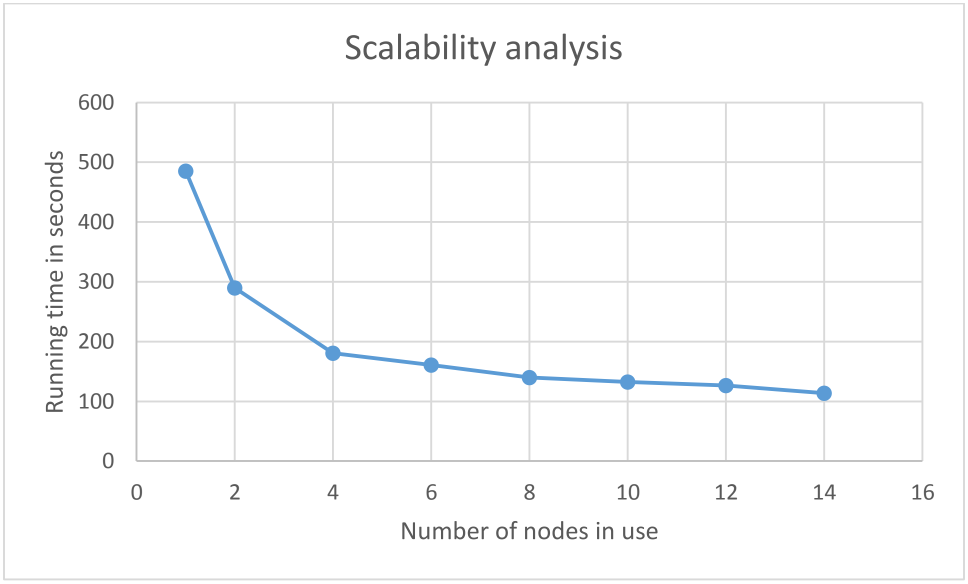

Figure 9. Scalability analysis

5. CONCLUSIONS AND FUTURE WORK

Hadoop is the choice of platform for large-scale data processing, its good scalability, high availability and fault tolerance gives the industry an approach to retrieve information from big data in a reasonably short amount of time. The MapReduce programming model is easy to use, people can create MapReduce programs without a detailed understanding of the internal system design of Hadoop. Apart from Java, now Hadoop streaming allows people to program MapReduce jobs in other languages like R, Python and etc. A number of applications are built on top of Hadoop framework to even more advance the large-scale data processing technology.

Machine learning is a field combined with statistics and artificial intelligence to learn from data and discover the hidden but valuable information behind data. In the Big Data era, machine learning is extremely useful for enterprises to turn the “garbage data” to valuable information and help them to make better business decisions. Researchers have done a lot of investigation to distribute machine learning algorithms to expedite the computation process. A number of machine learning algorithms have been incorporated into Mahout [22], a Java scalable machine library.

In our study, we built a Naïve Bayes classifier by leveraging the MapReduce function and also perform a scalability analysis to see the relationship between the running time and the size of the cluster. It turns out that, without reducing the accuracy, a distributed Naïve Bayes classifier has a much higher performance compared with the running of the algorithm on a single machine. A MapReduce version of the Naïve Bayes classifier turns out to be extremely efficient when dealing with large amount of data.

Our work in this paper is only a small step towards leveraging machine learning algorithms using the MapReduce model. In the future, one direction can be experimenting with more machine learning algorithms using MapReduce and using Mahout API to benchmark our experiments. The other option could be to understand more internal system design of Hadoop framework so to better utilize our cluster resources for the job we created, for example, investigating the optimal number of mappers and reducers used for a job to maximize the throughput of our large-scale data processing jobs.

6. REFERENCES

[1] J. Bughin, M. Chui, and J. Manyika. 2010. "Clouds, big data, and smart assets: Ten tech-enabled business trends to watch". *McKinsey Quarterly 56(1), pp. 75-86*.

[2] R. Kohavi, L. Mason, R. Parekh, and Z. Zheng. 2004. "Lessons and challenges from mining retail e-commerce data" *Machine Learning 57(1-2), pp. 83-113*.

[3] K. Chard, S. Caton, O. Rana, and K. Bubendorfer. 2010. "Social cloud: Cloud computing in social networks." *In IEEE 3rd International Conference on Cloud Computing, pp. 99-106*.

[4] R. Bryant, R. H. Katz, and E. D. Lazowska. 2008. "Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science and Society". *In Computing Research Initiatives for the 21st Century. Computing Research Association.*

[5] Zaslavsky, C. Perera, and D. Georgakopoulos. 2013. "Sensing as a service and big data". *International Conference on Advances in Cloud Computing, pp. 21-29*.

[6] K. McKelvey, A. Rudnick, M. D. Conover and F. Menczer. 2012. "Visualizing communication on social media: Making big data accessible". *arXiv preprint arXiv:1202.1367*.

[7] P. Zikopoulos and C. Eaton. 2011. "Understanding big data: Analytics for enterprise class hadoop and streaming data". *McGraw-Hill Osborne Media*.

[8] S. Madden. 2012. "From databases to big data". *IEEE Internet Computing, 16(3), 0004-6*.

[9] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money. 2013. "Big data: Issues and challenges moving forward.". *In System Sciences (HICSS), 46th Hawaii International Conference on (pp. 995-1004). IEEE.*

[10] P. Laube. 2014. "Grand Challenges in Computational Movement Analysis". *In Computational Movement Analysis (pp. 81-87). Springer International Publishing*.

[11] J. Dean and S. Ghemawat. 2004. “MapReduce: simplified data processing on large clusters”. *Communications of the ACM, 51(1), pp. 107-113*.

[12] S. Ghemawat, H. Gobioff and S. Leung. 2003. “The Google file system”. *Proceedings of the nineteenth ACM symposium on Operating systems principles*.

[13] P. Zikopoulos. 2012. “Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data”. *New York: McGraw-Hill*.

[14] Z. Guo, G. Fox and M. Zhou. 2012. “Investigation of Data Locality in MapReduce”. *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*.

[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. 2010. “The hadoop distributed file system”. *In Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on (pp. 1-10). IEEE*.

[16] T. Guzella and W. Caminhas. 2009. “A review of machine learning approaches to Spam filtering”. *Expert Systems With Applications 36(7), pp. 10206–10222*.

[17] W. Yih, J. Goodman and V. Carvalho. 2006. “Finding advertising keywords on web pages”. *Proceedings of the 15th international conference on World Wide Web*.

[18] E. Rosten and T. Drummond. 2006. “Machine learning for high-speed corner detection”. *Proceedings of the 9th European conference on Computer Vision - Volume Part I*.

[19] P. Baldi and S. Brunak. 1998. “Bioinformatics: The machine learning approach”. *Cambridge, Mass: MIT Press*.

[20] C. Chu, S. Kim, Y. Lin, Y. Yu, G. Bradski, A. Ng and K. Olukotun. 2006. “Map-Reduce for Machine Learning on Multicore”. *NIPS*.

[21] D. Gillick, A. Faria and J. DeNero. 2006. “MapReduce: Distributed Computing for Machine Learning”. *Berkeley Dec*.

[22] S. Owen, R. Anil, T. Dunning and E. Friedman. 2011. “Mahout in action”. *Manning*.

[23] A. K. Jain and M. N. Murty. 1999. “Data clustering: A review”. *ACM Computing Surveys, 32(3), pp. 264-323*.

[24] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu. 2002. "An efficient k-means clustering algorithm: Analysis and implementation". *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(7), pp. 881-892*.

[25] I. Gath and A. B. Geva. 1989. "Unsupervised optimal fuzzy clustering". *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 11(7), pp. 773-780*.

[26] Y. F. Yu and A. W. Zhou. 2011. "An Improved Algorithm of DBSCAN". *Computer Technology and Development, 21(2), pp. 30-33*.

[27] V. N. Vapnik. 1999. "An overview of statistical learning theory". *Neural Networks, IEEE Transactions on, 10(5), pp. 988-999*.

[28] J. Friedman, T. Hastie and R. Tibshirani. 2000. "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". *The annals of statistics, 28(2), pp. 337-407*.

[29] K. Muller, S. Mika, G. Ratsch, K. Tsuda and B. Scholkopf. 2001. "An introduction to kernel-based learning algorithms". *Neural Networks, IEEE Transactions on, 12(2), pp. 181-201*.

[30] S. C. Larson. 1931. "The shrinkage of the coefficient of multiple correlation". *Journal of Educational Psychology 22(1), pp. 45-55*.

[31] S. Arlot and A. Celisse. 2010. "A survey of cross-validation procedures for model selection". *Statistics surveys 4, pp. 40-79*.

[32] "UCI Machine Learning Repository" [Online]. Available: http://archive.ics.uci.edu/ml/. [Accessed 09/01/2014]

[33] T. G. Addair, D. A. Dodge, W. R. Walter and S. D. Ruppert. 2014. "Large-scale seismic signal analysis with Hadoop". *Computers & Geosciences 66, pp. 145-154*.

[34] X. Liu, J. Han, Y. Zhong, C. Han and X. He. 2009. "Implementing WebGIS on Hadoop: A case study of improving small file I/O performance on HDFS". *Cluster Computing and Workshops, IEEE International Conference on*.

[35] N. Mohandas, and S. M. Thampi. 2011. "Improving Hadoop performance in handling small files". *Advances in Computing and Communications 193, pp. 187-194*.

[36] "Hadoop Wiki - Partitioning your job into maps and reduces" [Online]. Available: http://wiki.apache.org/hadoop/HowManyMapsAndReduces. [Accessed 09/01/2014]

[37] D. Keren. 2002. "Painter identification using local features and naive bayes". *In Pattern Recognition Proceedings. 16th International Conference on (Vol. 2, pp. 474-477). IEEE*.

[38] A. Gray, M. Wingate, C. T. Davies, E. Gulez, G. P. Lepage, Q. Mason, M. Nobes and J. Shigemitsu. 2005. "B-meson decay constant from unquenched lattice QCD". *Physical review letters, 95(21), 212001*.

[39] T. White. 2012. "Hadoop: The definitive guide". *Yahoo Press*.

[40] "Hadoop Wiki - SequenceFile" [Online]. Available: http://wiki.apache.org/hadoop/SequenceFile. [Accessed 09/01/2014]

[41] S. B. Kim, K. S. Han, H. C. Rim and S. H. Myaeng. 2006. "Some effective techniques for naive bayes text classification". *Knowledge and Data Engineering, IEEE Transactions 18(11), pp. 1457-1466*.

[42] R. McCreadie, C. Macdonald and I. Ounis. 2011. "MapReduce indexing strategies: Studying scalability and efficiency". *Information Processing and Management 48(5), pp. 873-888*.