``

PROJECT REPORT

CSE 572 Data Mining

CLASSIFICATION

Spring 2013

SUBMITTED BY

Kedar Joshi(1205110916)

Akshaykumar Vaid(1205845923)

Introduction:

Classification is a classic data mining task, with roots in machine learning. The given problem statement is to classify unknown test records based on the training records with known class labels. The "Classification Problem" involves data which is divided into 20 groups, or classes labelled 1 to 20. The data mining classification model is to be constructed to predict the labels of a new example. So, we train the model using training records. We then ask the model to predict the class labels or groups of the unknown data. Of course, to ensure we can trust the predictions, there is generally a testing or validation stage as well.

Literature Review:

There are many possible solutions to this standard classification problem. Some of the most widely used techniques are as follows:

1. Decision tree classification:
   A decision tree classifier uses a tree like graph or model of decisions and their possible consequences to help identify a strategy to reach a goal. It is commonly used in operations research, specifically in decision analysis.
2. Naïve Bayes classification:
   This is a simple probabilistic classifier based on applying Bayes theorem with strong independence assumptions. This approach is particularly useful when the dimensionality of the data is high.
3. Support Vector Machines:
   SVM constructs a hyperplane or a set of hyperplanes in a high dimensional space, which can be used for classification. SVMs can efficiently perform non-linear classification using what is called kernel-trick, implicitly mapping their mapping their inputs into high dimensional feature spaces.
4. Nearest Neighbor classification:
   This is a method for classifying objects based on closest training examples in the feature space. It is a type of instance based learning or lazy learning where the function is only approximated locally and all computation is deferred until classification.
5. Rule Based classification:
   It is a technique for classifying records using a collection of “if..then..” rules. It is well suited for handling data sets with imbalanced class distributions.

These classification methods are already present in some of the tools like Weka, Oracle Data Mining (ODM).

Approach:

Required Software: PostgreSQL 9.2 Database, Eclipse IDE for Java

Pre-processing of data –

a. Read the input training data file and analyze the attributes. Remove all the attributes which are not present in any of the records. For example: If attribute id 150 is not present in any of the records, then we do not consider this attribute while building our model since the presence/absence of this attribute does not impact on label prediction.

b. Identify the distinct values of each attribute and count the number of occurrences of the same attribute id-attribute value pair for each class label. This information will be later used while calculating the probabilities.
c. Read the file produced in above step and append the class labels of each record next to each line.
d. Insert the data produced by the step above into database so that it can be later accessed. The columns of the training data table are – record_id, attr_id, attr_value, and label.

Model Building using Naïve Bayes Classifier:

1. Each data sample is represented by an n dimensional feature vector, X = (x1, x2 .... xn), depicting n measurements made on the sample from n attributes, respectively A1, A2, …., An.

2. Suppose that there are m classes, C1, C2.....Cm. Given an unknown data sample, X (i.e., having no class label), the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naive probability assigns an unknown sample X to the class Ci if and only if P (Ci/X) >P (Cj/X) for all 1< = j< = m and j! = I. Thus we maximize P (Ci|X). By Bayes theorem,

$$P(Ci/X) = (P(X/Ci)\ P(Ci))/P(X)$$

3. As P(X) is constant for all classes, only P (X|Ci) P (Ci) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, i.e. P (C1) = P (C2) =.....= P (Cm), and we would therefore maximize P (X|Ci). Otherwise, we maximize P (X|Ci) P (Ci). Note that the class prior probabilities may be estimated by P (Ci) = si/s, where Si is the number of training samples of class Ci, and s is the total number of training samples.

Pseudo-code:

Calculate label = C1, label = C2, label = C3….. , label = C20 probabilities from training input.

```
For Each Test Input Record
        For Each Attribute
                For Each distinct value of Attribute
                        Get the m-estimate value for this attribute and this value for each class.
                        Calculate Probabilities for each class label using Bayes theorem
                End For
        End For
Assign class label based on the maximum m-estimate value calculated above
End For
```

ProbabilityCalculator.java

This class calculates the probabilities for each distinct attribute and each distinct value of it per class. These are stored in a database and will be used while label prediction of test data. Note that in Naïve Bayes classification formula mentioned above, we are multiplying the probability values. So if any of the attribute values is missing, the whole equation will be equal to 0. To avoid this, we use m-estimate approach to replace the 0 values. The class MEstimator.java will update the 0 values with m-estimate values based on class prevalence and label count of each class.

InsertTestData.java

This class only inserts the lines of the test file into the database as is. We then use this table called "TestData" for further processing. The columns of this table are record_id, attr_id, attr_value and label.

ProcessTestData.java

Test data records are retrieved from the database and then – for each distinct attribute id and attribute value combination, we retrieve the probabilities for each class from the table populated in above steps. Probabilities for all classes are calculated assuming independence assumption between the attributes. Using Bayes theorem, the class for which the probability is highest, is assigned to the corresponding record.

Challenges:

1. Choice of database: If the input data is used directly into the program, it will start consuming a lot of program memory reducing its efficiency. Also it becomes very tedious to handle the file operations multiple times. With a relational database, retrieval of data becomes easier by using various methods such as Select queries etc.

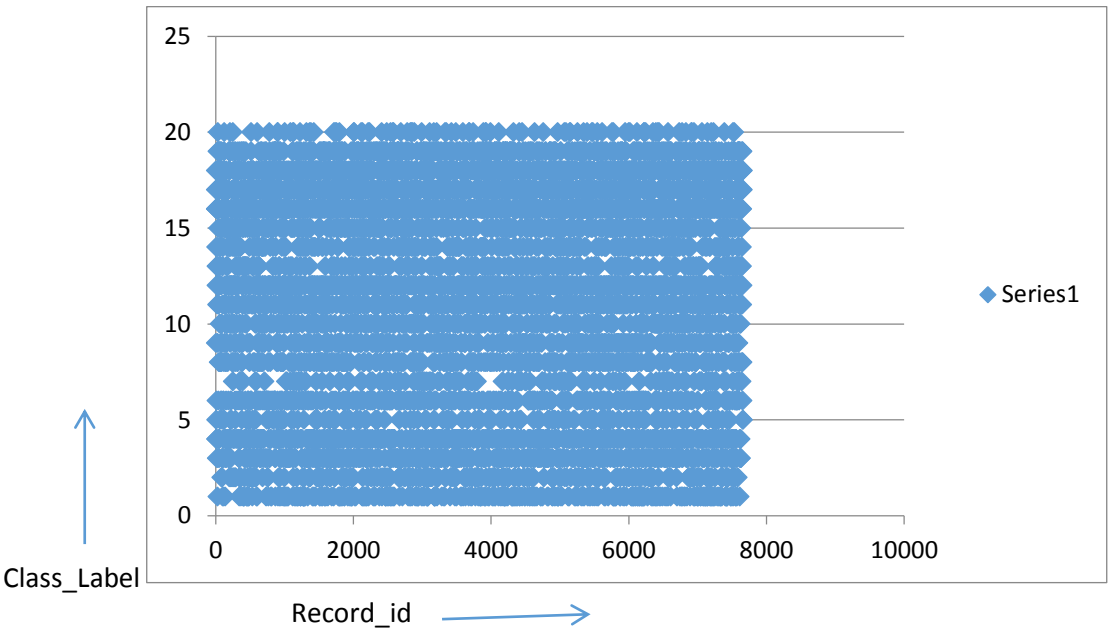
2. Storage of data: No database supports such a high dimensional data. The biggest challenge was to come up with an idea so that the data can be stored in the database. To overcome this, the data was stored as is in the file and then retrieved using efficient queries.

3. Feature Selection: Out of such a large number of features, the decision to ignore certain attributes was challenging. Ultimately, the attributes which did not appear in any of the records were eliminated.

4. Handling precision values during calculations of probabilities: It is observed that the proability values were very small (of the order of 10^{-5} or less). Hence while using the Bayes formula, the result was multiplied by a constant factor (around 10^{3}) so that the code can handle these small values.

Note: Please refer to **readme** file for steps to execute the code

Graph of record_id vs Class Labels



Graph demonstrating Record count of class labels