CSCI-B 565 DATA MINING
Analysis of Fraudulent sales data using Naive Bayes classification and K-means clustering algorithm
Computer Science Core

Fall 2012

Indiana University

Jayesh Kawli
jkawli@indiana.edu

11/26/2012

All the work herein is solely mine.

(a) **Analysis of Original data**

i. Given sales data represents real world sales scenario where database containing more than 4 Lakh records is given.This data actually represents real world scenario in the sense that, it is actually taken from existing anonymous sales company.

ii. It enlists the sales records which include attributes such as id of the salesman who sold that product,product id , total quantity and total sales value.
Given dataset contains more than 4 Lakh records. So from the real world scenario this is indeed a large dataset. Having very large dataset allows to have more training records thus we can make more efficient and workable classification model by learning provided with large training dataset.(This dataset contains more than 15000 training records excluding missing records)

iii. However eliminating NA records, only 15546 records have known labels. Remaining 371464 records have unknown label. Our task is to find label for this unknown records. Either ok or fraud

iv. Thus we use 15546 number of training records to train our Naive Bayes model and then use this model to infer labels for remaining (unknown) records.

v. Technically speaking given dataset contains 6 attributes. But first attribute which being id as unique identifier for each row does not have enough splitting/classification power. Thus we eliminate this column in our final data analysis. Also final column is nothing but the classifier for the specific record. This attribute is only used for data classification purpose based on values of remaining attributes

vi. First two columns represent salesman id and id of the product sold by respective salesmen.
In order to build training model for these non numeric attributes we use m-estimate approach where conditional probability is always non-zero as we always add some constant value to final evaluated value of conditional probability.

vii. Remaining two attributes are product quantity and total sell value. Since these are numeric values represent continuous distribution,the m-estimate approach used for first two attribute does not hold here as it is applicable to only discrete attributes. We use probability density function to calculate the probability for occurrence of each these numeric value for specific class label using training records

viii. Going through data, we found that there is correlation between product id, total value and quantity. For specific product we analysed all the records with ok label and evaluated the range of cost of individual unit of that product. Now knowing the range of individual cost for that product we can say, all the records for that particular product id which fall in that range correspond to ok labels while those deviate by large amount (Sold inappropriately) come under fraud label.

ix. Thus there is strong correlation between range of individual product id, their value and quantity thus sold. At the same time we make analysis of how many fraud and ok records associated with each salesman id. Higher is the number of fraud records associated with that salesman, higher is the probability that salesman is fraud for other products too. When Naive Bayes locates this salesman id in test record, it has high likelihood that it will give that salesman more bias to be classified as fraud instead of ok. Obviously, final result will depend on other attributes too

x. This is what we call as supervised learning. Since we learn labelling rules from training data and then apply them to test dataset. since this is kind of approach where result on test dataset hugely depends on training dataset. If the model fits too better then we can have model over fitting problem.

xi. Naive Bayes is an excellent classification algorithm in sense that it has got very good learning capability as evident from given example. Though, there are only 15000 training records and more than 3.7 Lakh test records. It gives impressive classification results

We are given raw fraudulent sales data. Some records are labelled (either ok or fraud) while others are unlabelled Our task is to use records with given labels as training data and then use this information to infer labels for unknown data

Going by statistics, I wrote C++ code to calculate following statistics and tested on original dataset thus provided.

When NA records were eliminated

Total Number of input records = 401146
Total number of NA records = 14136
Total number of records with all known attribute values = 387010

Total number of training records = 15546
Total number of training records with ok label = 14347
Total number of training records with fraud label = 1199
Fraction of records with missing values = (14136)/(401146)=0.035

Thus, very small fraction of records have missing values in it

Thus when calculating labels for unknown records amount of missing data is not very significant.
Which has also been verified experimentally in PPV calculation

PPV values calculation using training records

Total number of training records = 15732
Total number of records with NA values = 189

Effect of considering NA values in PPV calculation on training data

Average PPV value with NA records removed = 0.90772

Average PPV value with NA record values replaced by 0 = 0.904005

Thus from above analysis, it is concluded that considering or eliminating NA values in training data does not affect final value of average PPV by significant amount.

Thus from above discussion and very small value of ratio of (missing/total)=(189/15732)=0.012 (for training) and value of 0.035 (For whole data including labelled and unlabelled records) we conclude that amount of missing data is not significant

Thus for given data, we treat NA values as unacceptable and as a part of data cleaning process eliminate those rows with at least one missing attribute value.

(Though not evident from above results, it is possible that replacing NA values with any arbitrary unknown value essentially means we are assigning unknown values to some of the records which may not necessarily reflect what the original value that attribute possessed. This is applicable for any record with missing value in it. Thus to avoid probable skew related to records with missing value, we better chose to eliminate whole records with at least one missing values in it)

(b) As observed from given data, it contains more that one NA records. There are two options one is to eliminate NA records and other is to replace NA values with numeric 0

Initially there are 14136 records with at least one NA value of one of its attributes

NA values represent the missing data. Either we can eliminate them as there is no way to predict value of missing data or in other approach we can simply eliminate records with NA values. However this approach might cause trouble in final performance as along with NA values, we are also eliminating values of remaining attributes too.

In former approach replacing NA values with 0 might cause inconsistency in final data representation and hence we assess the PPV performance by replacing NA values with zero and completely eliminating rows with NA records

PPV values calculation using training records

Total number of training records = 15732
Total number of records with NA values = 189

Effect of considering NA values in PPV calculation on training data

Average PPV value with NA records removed = 0.90772
Average PPV value with NA record values replaced by 0 = 0.904005

Thus from above analysis, it is concluded that considering or eliminating NA values in training data does not affect final value of average PPV by significant amount (As tested on training dataset) - We choose to eliminate those rows with missing attribute values as part of data cleaning process

(c) Data mining algorithm employed.

i. Naive Bayes Classification algorithm

A. We are given very large data related fraudulent sales. This data is raw and some of the records might contain missing values for one or more attributes. Based on performance analysis, as a part of data cleaning we eliminate missing value. Now we have almost 15000 training records for which we know correct labels. Other than that we have more than 3,70,000 records which does not have any label and we need to find labels for this test dataset using model built on training dataset.

B. We implemented naive Bayes algorithm to train our model and ran the same model to infer labels for test dataset with unknown labels. While classifying unknown records, we use m-estimate approach where we eliminate the possibility of zero probability if any of the discrete attribute/label pair is missing in training data set. Using m-estimate, we still get non-zero probability value as long as value of final conditional probability for that attribute is concerned.

C. Checking the correctness of training model
We also tested the correctness of our training model. This procedure was performed on training dataset where labels of all records are known. We used 10-fold cross validation where given training dataset is divided into 10 partitions. Each partition is used as training dataset 9 times and test dataset once. We calculate labels of test dataset using remaining 90% training dataset. Then we compare final labels of test data with actual labels Matching records correspond to true positives and mismatched records correspond to false positives

Accuracy for single run of 10 fold cross validation = (Number of TP)/(Number of TP+Number of FP)

Overall accuracy for 10-fold cross validation = $\sum_{i=1}^{10} Accuracy_i$

ii. K-means classification algorithm

A. Once we get classified records with Naive Bayes algorithm next step is to use K-means clustering algorithm to divide data in two partitions namely, one partition represents ok data points and other as fraud data points. We then calculate PPV value for this classification using labels of each data points in respective clusters.

B. This value helps to assess the performance of Naive Bayes classification as well as K-means clustering algorithm.

C. Experimental PPV values for both Naive Bayes and K-means clustering implementation are illustrated in following sections

iii. Random forest classification algorithm

We used R package randomForest to analyze given fraudulent sales data. We used randomForest to classify data. We also used it to test the accuracy of classification model built using partial training data and treating rest of the data as test dataset. We then predict labels for test dataset and compare them with observed labels of either ok/fraud. Random forest algorithm gives detailed description of how many records are classified correctly and how many of them are classified incorrectly

Sections following show the analysis on classified data as well as analysis on training dataset to calculate the accuracy of classification

(d) Analysis of results

i. Naive Bayes Classification algorithm

Following considerations were given while implementing Naive bayes data classification algorithm

A. Used m - estimate approach.
Sometimes while finding labels for unknown data. We might get zero value for conditional probability

For example

For attribute attri and class cls, Naive Bayes without m-estimate approach uses formula

p(attri/cls)=$n_{attriandcls}/n_{cls}$

Where $n_{attriandcls}$=Number of records with attribute attri and class cls
and n_{cls}=Number of records with class cls

In case $n_{attriandcls}$ =0, probability of that test record with class cls becomes zero. To avoid this scenario of zero probability, we introduce notion of m-estimate while predicting labels for unknown data using formula

p(attri/cls)=$n_{attriandcls} + numsam * prioprob/n_{cls} + numsam$

Where $numsam$=Number of extra samples assumed with attribute attri and class cls for which no training record exists

prioprob=prior probability that unknown dataset might belong to class with label cls (Assumed to be 0.5 to avoid bias towards any of the specific class label)

B. Significance of m-estimate approach is that if any of the attribute-class combination is not present in training dataset then we avoid zero-probability scenario by assuming training samples of that class-attribute pair and adding constant value to avoid making conditional probability value to zero.

   Training sample size is assumed to be equal to 5

   Prior-probability value assumed to be = 0.5

C. Since both of these values are added to probability calculation for both ok and fraud classes, they equally weigh outcome for ok and fraud classes for given combination of attributes. Out main concern is to avoid zero valued conditional probability

D. Analysis on labelling of unclassified records

   Total Number of original records excluding records with missing values = 381070
   Total number of records with unknown labels = 371464
   Total number of training records with label OK= 14347
   Total number of training records with label fraud = 1199

E. Final Results

   Equal prior probability for class ok and fraud

   Prior estimate of probability for class OK=0.5
   Prior estimate of probability for class Fraud=0.5
   Total number of assumed training samples =5
   Total number of unknown records classified as OK = 370362
   Total number of unknown records classified as fraud = 1102
   Fraction of records classified OK= 0.997
   Fraction of records classified fraud= 0.003

F. Bias towards class fraud

   Prior estimate of probability for class OK=0.3
   Prior estimate of probability for class Fraud=0.7
   (Assuming constant sample size of 5)

Total number of unknown records classified as OK = 369150
Total number of unknown records classified as fraud = 2314
Fraction of records classified OK= 0.99377
Fraction of records classified fraud= 0.0063

G. **PPV values calculation -** Measuring accuracy of Naive Bayes classification model using 10 fold cross validation method. We divide given training data in 10 partitions. Now each partition will act as test dataset once and training dataset 9 times in overall 10 iterations
In each iteration we calculate class labels for test dataset and compare it with actual known class labels

Results:

Sample size=5
Prior estimate of probability for both classes fraud and ok =0.5

Iteration	Test partition	False Positives	True Positives	PPV
1	1	46	1508	0.970399
2	2	80	1474	0.94852
3	3	132	1422	0.915058
4	4	174	1380	0.888031
5	5	114	1440	0.926641
6	6	207	1347	0.866795
7	7	148	1406	0.904762
8	8	195	1359	0.874517
9	9	157	1397	0.898997
10	10	188	1366	0.879022

Average PPV value = (sum of all PPV values)/(Number of partitions)=9.07272/10=0.907272

In percentage=90.7272

H. Evaluating accuracy using 2-fold cross validation method

We divide given training dataset in two partitions. Each partition is used as training and test dataset exactly once. Then we take average of PPV values evaluated on both partitions

Other experimental parameters remain same as those used for 10-fold cross validation

Iteration	Test partition	False Positives	True Positives	PPV
1	1	46	1508	0.970399
2	2	207	1347	0.866795

Average PPV value = (sum of all PPV values)/(Number of partitions)=0.918597

In percentage=91.8597

Which is higher than what we got using 10-fold cross validation

I. Effect of using Laplace smoothing formula
Using the same notations and constant variables used in earlier parts,

Formula now changed to,

p(attri/cls)=$n_{attriandcls} + 1/n_{cls} + numsam$

Iteration	Test partition	False Positives	True Positives	PPV
1	1	56	1498	0.960394
2	2	77	1477	0.95045
3	3	130	1424	0.916358
4	4	172	1382	0.8889318
5	5	112	1442	0.927928
6	6	200	1354	0.8713
7	7	147	1407	0.905405
8	8	183	1371	0.882239
9	9	154	1400	0.900009
10	10	183	1377	0.882692

Average PPV value using Laplacian correction = 90.9054

Average PPV value using m - estimate approach = 90.7272

Thus from above, experiments, it is obvious that formula used to avoid zero conditional probability situation does not affect final result by significant margin

J. Analysis of results without m-estimate approach. We assume that if particular record type is not present in training dataset then its probability becomes zero. Since we are not assuming any kind of record type by taking arbitrary value of sample size and prior probability for both ok and fraud class.

Since we have large amount of test dataset it is possible that any attribute combination might not be present in training dataset. Without using m-estimate approach we might get zero probability value which is not seen in training dataset.

We analyse the results obtained when executing Naive Bayes algorithm without m-estimate approach .

Sample size = 0

Iteration	Test partition	PPV
1	1	0.900001
2	2	0.943372
3	3	0.916358
4	4	0.90991
5	5	0.888674
6	6	0.927928
7	7	0.877091
8	8	0.913771
9	9	0.903475
10	10	0.884615

Average PPV value = 0.90377

Max PPV value = 0.943372

Min PPV value = 0.877091

K. Difference between Average PPV value obtained with and without m-estimate approach =
|(Avg PPV value with m-estimate approach) -(Avg. PPV value without m-estimate approach)|
=|0.90377-0.907272 |=0.003502

Thus we did not observe significant difference in average PPV value with or without using m-estimate approach

L. Thus we classified unknown data and found out labels for it. Since there is no way to decide whether labels are correct or wrong, we ran our code on training data and evaluated accuracy of classification model using 10-fold cross validation method

   As evident from above result, we performed 10 iterations where each of the partition acts as a test dataset exactly once

   Max PPV value = 0.970399
   Min PPV value = 0.874517
   Average PPV value = 0.907272

   Thus our model classifies the unknown data with almost 91% accuracy. Thus we have pretty strong confidence in classification level of our model
   Also when classification model was tested with 2-fold cross validation method,
   Max PPV value = 0.970399
   Min PPV value = 0.866795
   Average PPV value = 0.918597

   Given the training dataset of size 15546 and test dataset of size of more than 370000 records, Naive Bayes is very effective learner and makes classification model which can classify large amount of unknown data even when it is trained on dataset having relatively small size

   Average Computation time for data classification on unknown records (Taken over 10 runs) = 8.5 Seconds

   Average Computation time for PPV calculation for training records (Taken over 10 runs)= 0.380 Seconds

   (Both tested on silo.cs.indiana.edu)

ii. K-means clustering algorithm on classified data

A. After running Naive Bayes classifier and classifying data points as either fraud or OK, we applied K-means clustering algorithm to assess quality of result.

B. We randomly choose two centroids and partition data in two clusters. After partitioning we calculate PPV value for each cluster to assess the quality of algorithm implementation

C. Analysis of classified data using K-Means clustering algorithm

   Dataset size = 381070 records

Test no	Number of Iterations.	Threshold	Number of Partitions	PPV	Execution time(Seconds)
1	27	2	2	0.977	60.97
2	93	2	5	0.9787	287.62
3	29	1	2	0.97708	69.58
4	67	1	4	0.977582	185.28
5	290	10	10	0.96778	1534.34

D. Since K-means cannot deal with non-numerical attribute values, we replace all non-numerical values with numeric constant. So that these values do not contribute to K-means clustering process.

For example in given sales data, we replace these values by numerical constant 1. Thus they do not play any role when we perform centroid selection and partitioning.

E. String attribute values of class labels are replaced by numeric values for convenience purpose. They play essentially same role as the original attributes represented as string data type.
We use following mapping

ok -> 2
Fraud -> 4

F. It was found with K-means clustering analysis that, since input file contains very large number of records (almost 4 Lakhs).Centroid convergence takes long time which is evident from above table

G. However we get fairly good value of PPV. We tested this algorithm on final classified data.
Following table shows the effect of running K-means clustering on original training data where

Total number of input Records excluding those with missing attribute values =15546

Test no	Number of Iterations.	Threshold	Number of Partitions	PPV	Execution time(Seconds)
1	18	2	2	0.928325	1.65
2	64	2	5	0.936315	7.27
3	10	1	2	0.928325	1.69
4	60	1	4	0.934466	7.14
5	162	2	10	0.926073	26.35

Now besides considering, only two numerical attributes for given sales data, we also analyzed these attributes along with other third derived attribute namely cost per unit.calculated as

cost/unit quantity=(value)/(quantity) added as an additional attribute

We know total quantity of items sold and total sell cost. Thus we introduce another derived attribute as cost per unit as third numeric attribute. Now these three attributes are correlated with each other in logical sense. Thus when we calculate new PPV value, there should not be much deviation from what PPV values we calculated earlier.

Thus K-means clustering performed with third attribute, namely, cost per unit item quantity

Test no	Number of Iterations.	Threshold	Number of Partitions	PPV	Execution time(Seconds)
1	15	2	2	0.928349	1.65
2	41	2	5	0.936501	5.46
3	11	1	2	0.928349	1.49
4	36	1	4	0.916565	4.19
5	82	2	10	0.934888	14.52

Thus, from above experiment, it was observed that introducing third attribute which is derived from already existing attribute value and quantity

A. Reduces the number of iterations to converge
B. Reduces total time required to converge to final solution
C. Does not affect final PPV value by very large margin

Average difference between PPV values between experiment perfromed with 2 attributes and 3 attributes = 0.008852

Which is not very significant. It happens because all 3 attributes are related to each other. This is the reason introducing third attribute does not affect PPV value by visibly large margin. However, as observed one effect is clearly visible that algorithm converges faster for more number (Three) attributes than the previous approach where only 2 attributes were used

D. Thus we analysed the performance of K-means algorithm by clustering two types of dataset first being training dataset and other being whole dataset which contain all the records including training and the one classified with Naive Bayes implementation.

Also from the training dataset we get high value of PPV which is evident from above table

E. While testing for PPV value for K-means clustering algorithm, non-numerical values were replaced by numeric constant and non-numeric labels "ok" and "Fraud" were replaced by numeric 2 and 4 respectively

F. **Conclusion:** Thus we took raw fraudulent sales data and used Naive bayes algorithm to classify unknown records. When tested our model against training dataset using 10-fold and 2-fold cross validation method ,we evaluated the average value of PPV to be almost 0.91 (In percentage 91).

G. Classified data is then fed to K-means clustering algorithm to form a variable sized partitions with variable threshold size which determines convergence moment. This experiment was also successful with average PPV value above 0.95

H. K-means clustering algorithm was also run successfully on training dataset with pre-determined labels In this case also, we tested it against variable values cluster size and threshold value. PPV values were above 0.91

Appendix

1 Source code for Naive Bayes implementation using C++

```
#include<iostream>
#include <fstream>
#include <map>
//#include <conio.h>
#include <time.h>
#include <vector>
#include <string.h>
#include<stdlib.h>
#include<math.h>
#define PI 3.14159265359
#define E 2.71828182846
#define sqrt_pi 2.506628274631

/* Author - Jayesh Kawli */

using namespace std;
int main()
{
    int total_time=0;
    map<string,int> values_ok;
    map<string,int> values_fraud;
    int temp=0;
    int vec_pos=0;
    int vec_overall=0;
    char buffer[40];
    string string1;
    map<int,long> tempholder;
    vector< vector<string> > sales_data_vec;
    int count=0;
    int uni=0;
    ofstream getoutdata1,getoutdata,getoutdatatrain;

    /*File for storing inferred labels for unknown data */

    getoutdata.open("fr.txt");

    /* Output file with all records including ok, fraud and unknowns inferred */

    getoutdata1.open("allokfraud.txt");

    /* File which stores all ok and fraud data from given pool of records.
    Basically contains training records. Which will then be used for PPV
    calculation */

    getoutdatatrain.open("okfraudmod.txt");


    /* Input file containing all records including ok, unknown and frauds and NA s */

    ifstream salesdata("salesf_transform1.txt");

    int gt=0;
    int indi;
```

```
/* Read the sales data file line by line. Until it reaches end of records */

while(!salesdata.eof())
{
    indi=0;
    bool isunknown=false;
    vector<int> attributeholder;
    char *indiattri;
    salesdata.getline(buffer,40);
    char *pch = strtok (buffer,"\t");
    vec_pos=0;
    vector<string> temp;
    temp.clear();
    int test=0;
    while(pch!=NULL)
    {
        char temp12[10];
        strcpy(temp12,pch);
        temp.insert(temp.begin()+vec_pos,temp12);
        test++;
        if(!strcmp(pch,"NA"))
        {
            gt++;
            pch=0;
            indi=1;
            break;
        }
        vec_pos++;

         /* Records are delimited by tab space */

        pch=strtok(NULL,"\t");
    }

     /* Check to ignore records with missing values */

    if(indi==0)
    {
        sales_data_vec.insert(sales_data_vec.begin()+vec_overall,temp);
        vec_overall++;
    }
    else
    {
        indi=0;

    }
}

vector< vector<string> >::iterator it1;
vector<string>::iterator it2;

int number_ok=0,number_fraud=0,number_unknown=0;
int quantity_fraud=0,quantity_ok=0,value_ok=0,value_fraud=0;

/* Store the count for each value in attribute column
salesman and product id for both class labels ok and fraud
Store the count. And this will further be used to calculate
probabilites which will be used to find labels for test data
*/
```

```
 for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
 {

     /* If the record label is ok, then add values of quanity and values
     in appropriate variables this will then be used to calculate normalized
     value for each numerical attributes */

     if(!(((*it1).at(4).compare("ok"))))
     {
         quantity_ok+=atoi((*it1).at(2).c_str());
         value_ok+=atoi((*it1).at(3).c_str());
         number_ok++;
     }

     /* If the record label is fraud, then add values of quanity and values
     in appropriate variables this will then be used to calculate normalized
     value for each numerical attributes */

     else if(!((((*it1).at(4)).compare("fraud"))))
     {
         quantity_fraud+=atoi((*it1).at(2).c_str());
         value_fraud+=atoi((*it1).at(3).c_str());
         number_fraud++;
     }

     /* Unknown Labels. Do nothing */

     else
     {
         number_unknown++;
     }
 }

// cout<<"Number of Unknowns"<<number_unknown<<endl;
 //cout<<"Number of Oks"<<number_ok<<endl;
 //cout<<"Number of Frauds"<<number_fraud<<endl;
 //cout<<quantity_fraud/number_fraud<<"ohh  "<<quantity_ok/number_ok<<"  "
 <<value_ok/number_ok<<"  "<<value_fraud/number_fraud<<endl;

 /* Calculate mean values in each of the quantity and values columns for
 both ok and fraud labels. This is to calculate normalized values of
 numerical attributes and evaluate their probabilities for test/unknown data */

 double mean_quantity_fraud=(quantity_fraud/number_fraud);
 double mean_quantity_ok=(quantity_ok/number_ok);
 double mean_value_ok=(value_ok/number_ok);
 double mean_value_fraud=(value_fraud/number_fraud);

 double var_quantity_fraud=0,var_quantity_ok=0,var_value_ok=0,var_value_fraud=0;
 int flag=0;

 /* Once caluclate mean values, calculate variance using standard variance formula */

 for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
 {

     if(!(((*it1).at(4).compare("ok"))))
```

```
        {
            flag=1;
            var_quantity_ok+=pow((atoi((*it1).at(2).c_str())-mean_quantity_ok),2);
            var_value_ok+=pow((atoi((*it1).at(3).c_str())-mean_value_ok),2);
        }
        else if(!((((*it1).at(4)).compare("fraud"))))
        {
            flag=1;
            var_quantity_fraud+=pow((atoi((*it1).at(2).c_str())-mean_quantity_fraud),2);
            var_value_fraud+=pow((atoi((*it1).at(3).c_str())-mean_value_fraud),2);
        }
    }

    /* Divide by number of samples n. It might be n or (n-1) */

    var_quantity_fraud=(double)(var_quantity_fraud/number_fraud);
    var_quantity_ok=(double)(var_quantity_ok/number_ok);
    var_value_ok=(double)(var_value_ok/number_ok);
    var_value_fraud=(double)(var_value_fraud/number_fraud);

    /* Calculate standard deviation */

    double std_quantity_fraud=sqrt(var_quantity_fraud);
    double std_quantity_ok=sqrt(var_quantity_ok);
    double std_value_ok=sqrt(var_value_ok);
    double std_value_fraud=sqrt(var_value_fraud);

    double const1_quantity_fraud=(double)(1/(sqrt_pi*std_quantity_fraud));
    double const1_quantity_ok=(double)(1/(sqrt_pi*std_quantity_ok));
    double const1_value_ok=(double)(1/(sqrt_pi*std_value_ok));
    double const1_value_fraud=(double)(1/(sqrt_pi*std_value_fraud));

    double const2_quantity_fraud=(double)pow(E,-(double)(1/(2*var_quantity_fraud)));
    double const2_quantity_ok=(double)pow(E,-(double)(1/(2*var_quantity_ok)));
    double const2_value_ok=(double)pow(E,-(double)(1/(2*var_value_ok)));
    double const2_value_fraud=(double)pow(E,-(double)(1/(2*var_value_fraud)));

    /* Store all the records without missing values having
    labels as ok or fraud in another file.
    This file will then be used to calculate PPV values and
    estimate accuracy of Naive Bayes classifier model */

    for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
    {
        int y=0;

        if(!(*it1).at(4).compare("fraud") || !(*it1).at(4).compare("ok"))
        {
            while(y!=5)
            {
                /*                                if(y==0 || y==1)
                                                {
                                                  getoutdatatrain<<"1"<<"\t";
                                                getoutdata1<<"1"<<"\t";

                                                }
                                                else*/
                //   {
if(y==4)
break;
                getoutdatatrain<<(*it1).at(y)<<"\t";
                getoutdata1<<(*it1).at(y)<<"\t";
                // }
                y++;
            }
double ratio11=(double)(atoi((*it1).at(3).c_str())/atoi((*it1).at(2).c_str()));
getoutdatatrain<<ratio11<<"\t";
getoutdatatrain<<(*it1).at(4)<<"\t";
            getoutdatatrain<<endl;
            getoutdata1<<endl;
            continue;
        }

        for(int i=0; i<2; i++)
        {
            if(!((*it1).at(4).compare("unkn")))
            {
                break;
            }
            else if(!((*it1).at(4).compare("fraud")))
            {

                getoutdata1<<endl;
                if(values_fraud.count((*it1).at(i))>0)
                {
                    values_fraud[(*it1).at(i)]++;
                }
                else
                {
                    values_fraud[(*it1).at(i)]=1;
                }
            }
            else if(!((*it1).at(4).compare("ok")))
            {
                if(values_ok.count((*it1).at(i))>0)
                {
                    values_ok[(*it1).at(i)]++;
                }
                else
                {
                    values_ok[((*it1).at(i))]=1;
                }
            }
        }
    }

    map<string,int>::iterator it3;
    getoutdatatrain.close();
    double temp_value_fraud=0,temp_quantity_fraud=0,temp_value_ok=0,temp_quantity_ok=0;
    double prior_estimate=0.5;
    double sample_size=5;
    int tot_training=number_ok+number_fraud;
    int test_ok=0,test_fraud=0,equal=0;

    /* Tested with different values of prior estimate. There was little difference
    in final estimation of class labels for unknown data */

    double prior_estimate_fraud=0.5,prior_estimate_ok=0.5;
```

```
/* Start the clock once execution begins */

total_time=clock();

for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
{
    if(!((*it1).at(4).compare("unkn")))
    {

        temp_value_ok=pow((atoi((*it1).at(3).c_str())-mean_value_ok),2);
        temp_quantity_ok=pow((atoi((*it1).at(2).c_str())-mean_quantity_ok),2);
        temp_value_ok=(const1_value_ok)*pow((const2_value_ok),temp_value_ok);
        temp_quantity_ok=(const1_quantity_ok)*pow((const2_quantity_ok),temp_quantity_ok);
        double ok=(double)((double)(number_ok)/(number_ok+number_fraud)*((values_ok[(*it1).at(0)]+
        (sample_size*prior_estimate_ok))/(number_ok+sample_size))*((values_ok[(*it1).at(1)]+
        (sample_size*prior_estimate_ok))/(number_ok+sample_size))*temp_quantity_ok*temp_value_ok);


        temp_value_fraud=pow((atoi((*it1).at(3).c_str())-mean_value_fraud),2);
        temp_quantity_fraud=pow((atoi((*it1).at(2).c_str())-mean_quantity_fraud),2);
        temp_value_fraud=(const1_value_fraud)*pow((const2_value_fraud),temp_value_fraud);
        temp_quantity_fraud=(const1_quantity_fraud)*pow((const2_quantity_fraud),temp_quantity_fraud);

        double fraud=(double)((double)(number_fraud)/(number_ok+number_fraud)*((values_fraud[(*it1).at(0)
        (sample_size*prior_estimate_fraud))/(number_fraud+sample_size))*((values_fraud[(*it1).at(1)]+
        (sample_size*prior_estimate_fraud))/(number_fraud+sample_size))*temp_quantity_fraud*temp_value_fr

        int i=0;
        while(i!=4)
        {
            /*    if(i==0 || i==1)
                {
                        getoutdata<<"1"<<"\t";
                            getoutdata1<<"1"<<"\t";

                        }*/
            //  else
            //{
            getoutdata<<(*it1).at(i)<<"\t";
            getoutdata1<<(*it1).at(i)<<"\t";
            // }
            i++;
        }

        /* Once classified data, store all the details for that
        unknonw record in new file along with inferred label
        using training data and Naive Bayes classification algorithm */

        //cout<<ok<<"   "<<fraud<<endl;
        if(ok<fraud)
        {
            test_fraud++;
            getoutdata1<<"fraud";
            getoutdata<<"fraud";
        }
        else
        {
            test_ok++;
```

```
                getoutdata1<<"ok";
                getoutdata<<"ok";
            }

            getoutdata<<endl;
            getoutdata1<<endl;
        }
    }
    cout<<"Total time taken (In Seconds)   "<<((float)(clock()-total_time)/CLOCKS_PER_SEC)*2<<endl;
    cout<<"Number of unknown records classified as ok   "<<test_ok<<"  "<<endl;
    cout<<"Number of unknown records classified as fraud   "<<test_fraud<<"  "<<endl;
    cout<<"****************End of Naive Bayes****************"<<endl;
    getoutdata.close();
    getoutdata1.close();
    //getch();
    return 1;
}
```

2 Source code to evaluate accuracy of Naive Bayes classification model using training data

```
#include<iostream>
#include <fstream>
#include <map>
#include<iomanip>
//#include<conio.h>
#include<time.h>
#include <vector>
#include <string.h>
#include<stdlib.h>
#include<math.h>
#define PI 3.14159265359
#define E 2.71828182846
#define sqrt_pi 2.506628274631
using namespace std;

void naive_bayes(vector< vector<string> >,map<int,string>,int begin_index);
double ratio_final;

/* Author - Jayesh Kawli */

int main()
{
    /* Variables */

    int ppv_time=0;
    int vec_pos=0;
    int vec_overall=0;
    int gt=0;
    vector< vector<string> > sales_data_vec;
    char buffer[40];
    int indi;

    /* Input file containing training records with original class labels of OK or frauds */

    ifstream salesdata("okfraud.txt");
```

```
/* Reading training records from input file */

while(!salesdata.eof())
{
    indi=0;
    bool isunknown=false;
    vector<int> attributeholder;
    char *indiattri;
    salesdata.getline(buffer,40);
    char *pch = strtok (buffer,"\t");
    vec_pos=0;
    vector<string> temp;
    temp.clear();
    int test=0;
    while(pch!=NULL)
    {
        char temp12[10];
        strcpy(temp12,pch);
        temp.insert(temp.begin()+vec_pos,temp12);
        test++;

        /* Check to ignore records with missing values */

        if(!strcmp(pch,"NA"))
        {
            pch=0;
            gt++;
            indi=1;
            break;
        }
        vec_pos++;

        /* Records are delimited by tab space */

        pch=strtok(NULL,"\t");
    }
    if(indi==0)
    {
        sales_data_vec.insert(sales_data_vec.begin()+vec_overall,temp);
        vec_overall++;
    }
    else
    {
        indi=0;
    }
}

//cout<<"The Number of records with missing values  "<<gt<<endl;

int single_block_size =(int)ceil(sales_data_vec.size()/10);
double last_block_size=(double)(single_block_size+(sales_data_vec.size()%10));
int data=0;
map<int,string> label_holder;
ppv_time=clock();
int begin1,begin,iter=0;

for(int i=0; i<10; i++)
{
    begin1=i*(single_block_size);
    iter=0;

    /* Store all known values in loabel_holder buffer, replace those values
    with unkn label and then calculate labels for them */

    string vec_label;
    for(int j=(i)*single_block_size; j<=((i+1)*(single_block_size))-1; j++)
    {
        vec_label=sales_data_vec.at(j).at(4);
        sales_data_vec[j][4]="unkn";
        label_holder[iter++]=vec_label;

    }

    /* Following loop takes care of remaining elements at the last block.
    Remaining elements are added in vector and passed to naive Bayes algorithm
    for final classification verification */

    if(i==9)
    {
        int start_pos=(i+1)*(single_block_size);
        int span=(int)(last_block_size-single_block_size);
         for(int k=start_pos;k<start_pos+span;k++)
         {
                vec_label=sales_data_vec.at(k).at(4);
                sales_data_vec[k][4]="unkn";
                label_holder[iter++]=vec_label;
         }
    }

    /*Give values to Naive bayes with test and training dataset to calculate model accuracy*/

    naive_bayes(sales_data_vec,label_holder,begin1);
    begin=0;

    /* Replace all unknown values with their original labels
    after one pass. In the next pass, again replace lebels of
    next block with all unknown and store their true classes in another
    buffer */

    for(int j=(i)*single_block_size; j<(i+1)*(single_block_size); j++)
    {
        sales_data_vec[j][4]=label_holder[begin];
        begin++;
    }

    /* Always clear label_holder buffer when used for next round */

    label_holder.clear();
}

/* Gives the statistical measure of algorithm accuracy and performance.
Denotes execution time and average PPV value combined of all the
iterations */

cout<<endl<<"Total Execution time (In seconds) for PPV "<<(((float)ppv_time/CLOCKS_PER_SEC)*10)<<endl;
cout<<endl<<"Avergae PPV value for Naive Bayes algorithm for Sales data is: "<<(ratio_final/10)<<endl;
cout<<endl<<"Avergae PPV value for Naive Bayes algorithm for Sales data is (Percentage): "<<endl
```

```
    <<(ratio_final*10)<<endl;

    //getch();
    return 1;
}
void naive_bayes(vector< vector<string> > sales_data_vec,map<int,string> training_label,int begin_index)
{
    map<string,int> values_ok;
    map<string,int> values_fraud;
    int temp=0;
    string string1;
    map<int,long> tempholder;

    int count=0;
    int uni=0;
    ofstream getoutdata;

    int gt=0;

    vector< vector<string> >::iterator it1;
    vector<string>::iterator it2;

    int number_ok=0,number_fraud=0,number_unknown=0;
    int quantity_fraud=0,quantity_ok=0,value_ok=0,value_fraud=0;

    /* Store the count for each value in attribute column
    salesman and product id for both class labels ok and fraud
    Store the count. And this will further be used to calculate
    probabilites which will be used to find labels for test data
    */

    for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
    {
        /* If the record label is ok, then add values of quanity and values
        in appropriate variables this will then be used to calculate normalized
        value for each numerical attributes */

        if(!(((*it1).at(4).compare("ok"))))
        {
            quantity_ok+=atoi((*it1).at(2).c_str());
            value_ok+=atoi((*it1).at(3).c_str());
            number_ok++;
        }

        /* If the record label is false, then add values of quanity and values
        in appropriate variables this will then be used to calculate normalized
        value for each numerical attributes */

        else if(!((((*it1).at(4)).compare("fraud"))))
        {
            quantity_fraud+=atoi((*it1).at(2).c_str());
            value_fraud+=atoi((*it1).at(3).c_str());
            number_fraud++;
        }

        /* Unknown Labels. Do nothing */

        else
        {
```

```
        number_unknown++;
    }
}

/* Calculate mean values in each of the quantity and values columns for
both ok and fraud labels. This is to calculate normalized values of
numerical attributes and evaluate their probabilities for test/unknown data */

double mean_quantity_fraud=(quantity_fraud/number_fraud);
double mean_quantity_ok=(quantity_ok/number_ok);
double mean_value_ok=(value_ok/number_ok);
double mean_value_fraud=(value_fraud/number_fraud);
double var_quantity_fraud=0,var_quantity_ok=0,var_value_ok=0,var_value_fraud=0;

/* Once caluclate mean values, calculate variance using standard variance formula */

for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
{
    if(!(((*it1).at(4).compare("ok"))))
    {
        var_quantity_ok+=pow((atoi((*it1).at(2).c_str())-mean_quantity_ok),2);
        var_value_ok+=pow((atoi((*it1).at(3).c_str())-mean_value_ok),2);
    }
    else if(!((((*it1).at(4)).compare("fraud"))))
    {
        var_quantity_fraud+=pow((atoi((*it1).at(2).c_str())-mean_quantity_fraud),2);
        var_value_fraud+=pow((atoi((*it1).at(3).c_str())-mean_value_fraud),2);
    }
}

/* Divide by number of samples it might be n or (n-1) */

var_quantity_fraud=(double)(var_quantity_fraud/number_fraud);
var_quantity_ok=(double)(var_quantity_ok/number_ok);
var_value_ok=(double)(var_value_ok/number_ok);
var_value_fraud=(double)(var_value_fraud/number_fraud);

/* Calculate standard deviation */

double std_quantity_fraud=sqrt(var_quantity_fraud);
double std_quantity_ok=sqrt(var_quantity_ok);
double std_value_ok=sqrt(var_value_ok);
double std_value_fraud=sqrt(var_value_fraud);

double const1_quantity_fraud=(double)(1/(sqrt_pi*std_quantity_fraud));
double const1_quantity_ok=(double)(1/(sqrt_pi*std_quantity_ok));
double const1_value_ok=(double)(1/(sqrt_pi*std_value_ok));
double const1_value_fraud=(double)(1/(sqrt_pi*std_value_fraud));

double const2_quantity_fraud=(double)pow(E,-(double)(1/(2*var_quantity_fraud)));
double const2_quantity_ok=(double)pow(E,-(double)(1/(2*var_quantity_ok)));
double const2_value_ok=(double)pow(E,-(double)(1/(2*var_value_ok)));
double const2_value_fraud=(double)pow(E,-(double)(1/(2*var_value_fraud)));

for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
{
    for(int i=0; i<2; i++)
    {
        if(!((*it1).at(4).compare("unkn")))
        {
            break;
        }
        else if(!((*it1).at(4).compare("fraud")))
        {
            if(values_fraud.count((*it1).at(i))>0)
            {
                values_fraud[(*it1).at(i)]++;
            }
            else
            {
                values_fraud[(*it1).at(i)]=1;
            }
        }
        else if(!((*it1).at(4).compare("ok")))
        {
            if(values_ok.count((*it1).at(i))>0)
            {
                values_ok[(*it1).at(i)]++;
            }
            else
            {
                values_ok[((*it1).at(i))]=1;
            }
        }
    }
}

map<string,int>::iterator it3;

/*For each unknown records, calculate the probability of it being either ok
or fraud. Now based upon these probability values */

double prior_estimate_fraud;
double prior_estimate_ok;

/* Tested with different values of prior estimate. There was little difference
in final estimation of class labels for unknown data */

prior_estimate_fraud=0.5;
prior_estimate_ok=0.5;
double temp_value_fraud=0,temp_quantity_fraud=0,temp_value_ok=0,temp_quantity_ok=0;

/* Sample size determines, size of dummy sample assumed to be having that class label which
is not found is any of the test dataset. This is assumed to be 5 on experimental calculation */

double sample_size=5;
int iit=0;
int tr=0,fal=0;
int tty=0;

for(it1=sales_data_vec.begin(); it1!=sales_data_vec.end(); it1++)
{
    tty++;
    if(!((*it1).at(4).compare("unkn")))
    {
        temp_value_ok=pow((atoi((*it1).at(3).c_str())-mean_value_ok),2);
        temp_quantity_ok=pow((atoi((*it1).at(2).c_str())-mean_quantity_ok),2);
        temp_value_ok=(const1_value_ok)*pow((const2_value_ok),temp_value_ok);
```

```
            temp_quantity_ok=(const1_quantity_ok)*pow((const2_quantity_ok),temp_quantity_ok);
            double ok=(double)((double)(number_ok)/(number_ok+number_fraud)*((values_ok[(*it1).at(0)]+
            (sample_size*prior_estimate_ok))/(double)(number_ok+sample_size))*((values_ok[(*it1).at(1)]+
            (sample_size*prior_estimate_ok))/(double)(number_ok+sample_size))*temp_quantity_ok*temp_value_ok)

            temp_value_fraud=pow((atoi((*it1).at(3).c_str())-mean_value_fraud),2);
            temp_quantity_fraud=pow((atoi((*it1).at(2).c_str())-mean_quantity_fraud),2);
            temp_value_fraud=(const1_value_fraud)*pow((const2_value_fraud),temp_value_fraud);
            temp_quantity_fraud=(const1_quantity_fraud)*pow((const2_quantity_fraud),temp_quantity_fraud);

            double fraud=(double)((double)(number_fraud)/(number_ok+number_fraud)*((values_fraud[(*it1).at(0)
            (sample_size*prior_estimate_fraud))/(double)(number_fraud+sample_size))*((values_fraud[(*it1).at(
            (sample_size*prior_estimate_fraud))/(double)(number_fraud+sample_size))*temp_quantity_fraud*temp_

            int i=0;

            string final_res;

            /* This is actual testing with known and evaluated data
            When algorithm tells it is fraud and it matches with actual
            class, it increments true positive variable, else increments
            false positive label */

            if(ok<fraud)
            {
                if(training_label[iit].compare("fraud"))
                {
                    fal++;
                }
                else
                {
                    tr++;
                }
            }
            else if(ok>=fraud)
            {
                if(training_label[iit].compare("ok"))
                {
                    fal++;
                }
                else
                {
                    tr++;
                }
            }
            getoutdata<<endl;
            iit++;
        }
    }

    /* Now we have count of false and true positives.
    Calculate PPV values using standard formula  */

    double ratio=(double)(tr)/(tr+fal);
    ratio_final+=ratio;
cout<<setw(10);
    cout<<ratio<<"  Calculating Ratio.......iterating in loop...."<<endl;
//cout<<"*********************************************\n";
}
```