Meenakshi Nagarajan

Mini Project 4: Cloud Computing Labs: Privacy-Preserving Methods for Outsourced Computation

Part 1: Multiplication perturbation for outsourced data mining.

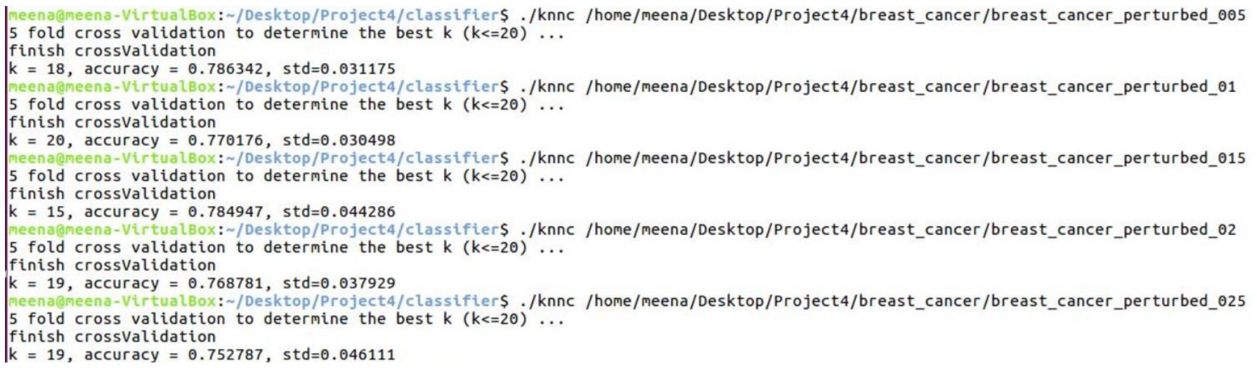
Question 1.1 Implement the GDP algorithm. Paste your code.

Solution:
```
package gdp;

import java.io.*;
import java.util.*;
import org.ejml.data.DenseMatrix64F;
import org.ejml.ops.RandomMatrices;

public class gdpalgorithm {

	public static void main(String [] args) throws Exception
	{
		Random random = new Random();
		//InputStream resource=gdpalgorithm.class.getResourceAsStream();
		SVMReader svmreader = new SVMReader(args[1]); //Reading original data
with svm reader
		int [] labels = svmreader.getLabels();
		int rows = svmreader.getnrows();
		int cols = svmreader.getncols();
		double mean=0.0;
		double data[] = new double[cols];
		double[][] records = svmreader.getRecords();
		float[] y = new float[cols];
		double[] t = new double[cols];
		double[] d = new double[cols];
		float sd = Float.parseFloat(args[0]);
		DenseMatrix64F R = RandomMatrices.createOrthogonal(cols,cols,random);
//Orthogonal matrix
		DenseMatrix64F T = RandomMatrices.createRandom(cols,cols,random);
//Random Secret vector matrix
		double[][] orthogonal = convertToArray(R); //array conversion
		double[][] t1 = convertToArray(T);
		FileWriter fw = new FileWriter(args[2],true);
		for (int i=0; i<rows;i++)
		{
			for (int j=0; j<cols; j++)
			{
				data[j] = records[i][j];
			}
		for (int k=0;k<cols;k++)
		{
			for (int j=0; j<cols; j++) //Rx
			{
				d[k]=0;
				d[k]+=orthogonal[k][j]*data[j];
			}
			}
			for (int k=0; k<cols; k++) //t
			{
				for (int j=0; j<cols; j++)
				{
					t[j] = t1[k][j];
				}
			}
			fw.write(labels[i]+" "); //y=Rx+t+d
			for (int j=0,k=1; j<cols; j++,k++)
			{
				y[j] =(float)((float)d[j] + (float)t[j] + 
(float)getGaussian(mean, sd*sd));
				fw.write(k+":"+y[j]+" ");
			}
			fw.write("\r\n");
			}
			fw.close();
			}
		private static double getGaussian(double mean, double sd){
			return mean + new Random().nextGaussian() * sd;
		}
		private static double[][] convertToArray(DenseMatrix64F matrix)
		{
			double array[][] = new double[matrix.getNumRows()][matrix.getNumCols()];
			for (int i=0; i<matrix.getNumRows(); i++)
			{
				for (int j=0; j<matrix.getNumCols(); j++)
				{
					array[i][j] = matrix.get(i,j);
				}
			}
			return array;
		}
}
```

Question 1.2 Conduct the accuracy evaluation on the three datasets and report your experimental results with the following table for each dataset. Each cell represents the accuracy and standard deviation (e.g., in the form 0.82+/-0.02) for the corresponding classification method and the sigma setting. Note that SVM output does not have standard deviation, which is fine. Interpret the table and write down your observations and conclusions.

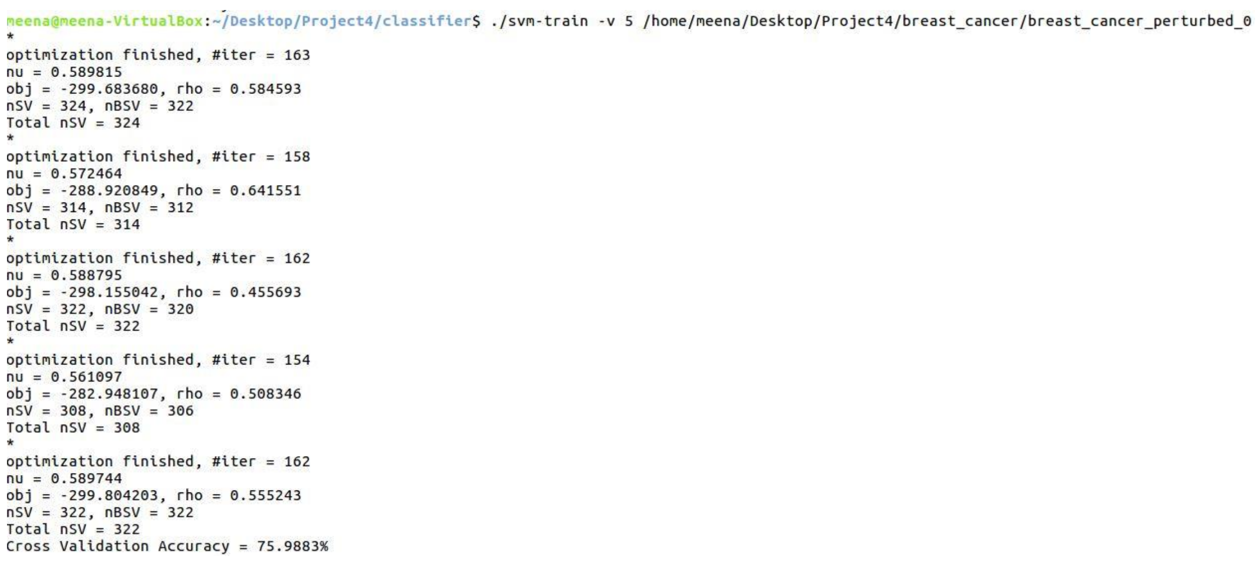
Solution:

kNN:



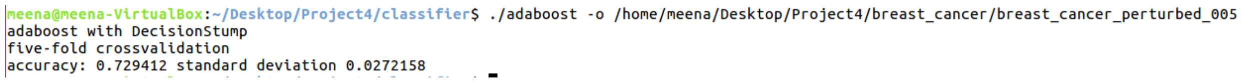
```
meena@meena-VirtualBox:~/Desktop/Project4/classifier$ ./knnc /home/meena/Desktop/Project4/breast_cancer/breast_cancer_perturbed_005
5 fold cross validation to determine the best k (k<=20) ...
finish crossValidation
k = 18, accuracy = 0.786342, std=0.031175
meena@meena-VirtualBox:~/Desktop/Project4/classifier$ ./knnc /home/meena/Desktop/Project4/breast_cancer/breast_cancer_perturbed_01
5 fold cross validation to determine the best k (k<=20) ...
finish crossValidation
k = 20, accuracy = 0.770176, std=0.030498
meena@meena-VirtualBox:~/Desktop/Project4/classifier$ ./knnc /home/meena/Desktop/Project4/breast_cancer/breast_cancer_perturbed_015
5 fold cross validation to determine the best k (k<=20) ...
finish crossValidation
k = 15, accuracy = 0.784947, std=0.044286
meena@meena-VirtualBox:~/Desktop/Project4/classifier$ ./knnc /home/meena/Desktop/Project4/breast_cancer/breast_cancer_perturbed_02
5 fold cross validation to determine the best k (k<=20) ...
finish crossValidation
k = 19, accuracy = 0.768781, std=0.037929
meena@meena-VirtualBox:~/Desktop/Project4/classifier$ ./knnc /home/meena/Desktop/Project4/breast_cancer/breast_cancer_perturbed_025
5 fold cross validation to determine the best k (k<=20) ...
finish crossValidation
k = 19, accuracy = 0.752787, std=0.046111
```

SVM:



```
meena@meena-VirtualBox:~/Desktop/Project4/classifier$ ./svm-train -v 5 /home/meena/Desktop/Project4/breast_cancer/breast_cancer_perturbed_0
*
optimization finished, #iter = 163
nu = 0.589815
obj = -299.683680, rho = 0.584593
nSV = 324, nBSV = 322
Total nSV = 324
*
optimization finished, #iter = 158
nu = 0.572464
obj = -288.920849, rho = 0.641551
nSV = 314, nBSV = 312
Total nSV = 314
*
optimization finished, #iter = 162
nu = 0.588795
obj = -298.155042, rho = 0.455693
nSV = 322, nBSV = 320
Total nSV = 322
*
optimization finished, #iter = 154
nu = 0.561097
obj = -282.948107, rho = 0.508346
nSV = 308, nBSV = 306
Total nSV = 308
*
optimization finished, #iter = 162
nu = 0.589744
obj = -299.804203, rho = 0.555243
nSV = 322, nBSV = 322
Total nSV = 322
Cross Validation Accuracy = 75.9883%
```

Adaboost:



```
meena@meena-VirtualBox:~/Desktop/Project4/classifier$ ./adaboost -o /home/meena/Desktop/Project4/breast_cancer/breast_cancer_perturbed_005
adaboost with DecisionStump
five-fold crossvalidation
accuracy: 0.729412 standard deviation 0.0272158
```

Breast_cancer:

Methods	***no perturbation***	***Sigma=0.05***	***Sigma=0.1***	***Sigma=0.15***	***Sigma=0.2***	***Sigma=0.25***
kNN	0.967862 ± 0.025930	0.786342 ± 0.031175	0.770176 ± 0.030498	0.784947 ± 0.044286	0.768781 ± 0.037929	0.752787 ± 0.046111
SVM	97.2182%	75.9883%	75.9883%	75.549%	73.6457%	72.9136%
AdaBoost	0.957353 ± 0.0317967	0.729412 ± 0.0272158	0.744118 ± 0.0423672	0.732353 ± 0.0635934	0.717647 ± 0.0505698	0.707353 ± 0.0240521

heart_scale:

Methods	***no perturbation***	***Sigma=0.05***	***Sigma=0.1***	***Sigma=0.15***	***Sigma=0.2***	***Sigma=0.25***
kNN	0.808364 ± 0.029127	0.748364 ± 0.023723	0.760000 ± 0.026722	0.748727 ± 0.026283	0.759273 ± 0.013489	0.755636 ± 0.018312
SVM	82.963%	76.2963%	76.2963%	76.2963%	76.2963%	76.2963%
AdaBoost	0.825926 ± 0.0500343	0.696296 ± 0.0662539	0.707407 ± 0.0619748	0.677778 ± 0.0834362	0.666667 ± 0.0774685	0.644444 ± 0.0546218

diabetes_scale:

Methods	***no perturbation***	***Sigma=0.05***	***Sigma=0.1***	***Sigma=0.15***	***Sigma=0.2***	***Sigma=0.25***
kNN	0.747368 ± 0.035836	0.657570 ± 0.038897	0.681066 ± 0.032400	0.656357 ± 0.032450	0.643284 ± 0.028462	0.668114 ± 0.036305
SVM	77.0833%	65.1042%	65.1042%	65.1042%	65.1042%	65.1042%
AdaBoost	0.755556 ± 0.0315491	0.656209 ± 0.0170437	0.645752 ± 0.0274976	0.635294 ± 0.0274976	0.645752 ± 0.0385011	0.637908 ± 0.0298085

From above tables, it could be concluded that the kNN classification method is more stable.

Question 1.3 The provided classification programs have hidden the details of "cross-validation". The default setting is set to five folds. Find the definition of cross-validation from the Web. Describe the procedure of cross-validation with your own words, and answer why we need cross-validation.

Cross-validation in statistics is a means to measure the predictive performance of a model.
In cross-validation, a dataset is divided into training and testing data. Training data is used to train the model and testing data is used to validate the performance of the model. For instance,

if default setting is set to five folds then a dataset is divided into 5 sub samples. Out of the 5 sub samples, single sample of data is chosen as training data with the remaining sub samples as testing data. Analysis is performed on training data and the validation is done with testing data. This process is repeated for 5 times as it is five-fold cross-validation. On each iteration, one sub sample is chosen as training data. Once all iterations are completed, the average precision is measured. Cross-validation is important to choose privacy-preserving models.

Question 1.4 Under what assumptions the security of the GDP perturbation method is guaranteed? What are the potential attacks? - discuss a few you can think of.

- It has been assumed that the data mining services will be provided by the service provider honestly.
- Also, the security is guaranteed under the assumption that provider might look at the data stored and processed on their platforms.

Potential attacks

- Independent component analysis attacks: With the knowledge of basic statistics and probability density function of each column the data could be reconstructed.
- Distance-based inference: GDP preserves the distance between the data points in the perturbed dataset. With the knowledge about the data, the attacker could possibly map the distance between the points and their images in the perturbed dataset.

Few other attacks are as follows:

- Naïve estimation attack
- Subset-based inference

Part 2: Crypto-index for range query processing

Question 2.1 Develop a query evaluation program, the usage of which is

crypto-index-query original_data transformed_data bin_encoding_file dimenion_i start end

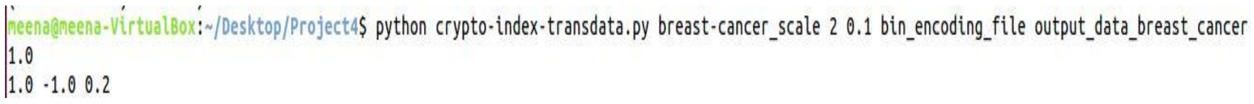
where [start, end] of dimension_i is the original range for search, bin_encoding is the one generated from the previous program, and transformed_data is the output_data from the previous program. The program will print the precision of the query, which is defined as (the number of records exactly in [start, end])/(the number of records returned by query processing on transformed data).

Program:

```
import os, sys, random,math
from readsvm import readSVM

def usage():
   print "python crypto-index-transdata.py input_data dimenion_i bin_width bin_encoding_file output_data"

if len(sys.argv)!=6:
   usage()
```

```
input_data = sys.argv[1]
dim = int(sys.argv[2])
bin_width = float(sys.argv[3])
enc_file = sys.argv[4]
output_data = sys.argv[5]

target, records = readSVM(input_data)

# dim starts from 0
vec_i = [rec[dim] for rec in records]
max_i = max(vec_i)
min_i = min(vec_i)
nbins = int (1.0/bin_width)
bin_size = float(max_i-min_i)/nbins
print max_i,min_i,bin_size
marks = [min_i+bin_size * i for i in range(nbins)]
marks.append(max_i)
binIDs = random.sample(xrange(10000000), nbins)
f= open(enc_file, "w+")
for i in range(nbins):
    print >>f, marks[i], marks[i+1], binIDs[i]
f.close()

f1=open(output_data, "w+")
#printing output data
for i in range(len(records)):
    binid = int(math.floor ((records[i][dim]-min_i)/bin_size))
    #print i, dim, binid, records[i][dim], min_i, bin_size, nbins, binIDs
    if binid == nbins: # upper bound
        binid -=1
    records[i][dim] = binIDs[binid]
    print >>f1,i, dim, binid, records[i][dim], min_i, bin_size, nbins, binIDs
f1.close()
```

Running the above program to get the outputfile and bin_encoding_file



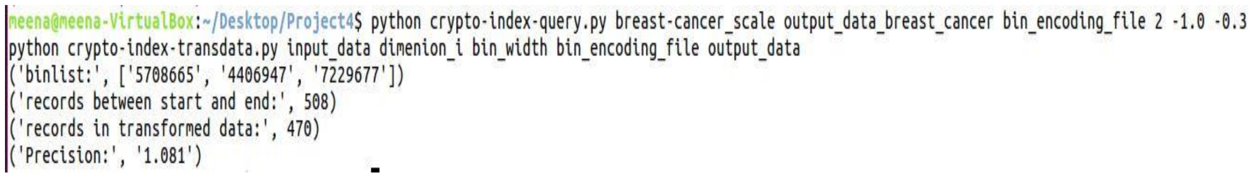
```
meena@meena-VirtualBox:~/Desktop/Project4$ python crypto-index-transdata.py breast-cancer_scale 2 0.1 bin_encoding_file output_data_breast_cancer
1.0
1.0 -1.0 0.2
```

#crypto-index-query

```
import os, sys, random,math
from readsvm import readSVM


def usage():
    print "python crypto-index-transdata.py input_data dimenion_i bin_width bin_encoding_file output_data"

if len(sys.argv)!=6:
    usage()

input_data = sys.argv[1]
dim = int(sys.argv[4])
start = sys.argv[5]
end = sys.argv[6]
enc_file = sys.argv[3]
transformed_data = sys.argv[2]
```

```
target, records = readSVM(input_data)
binlist=[]

#add binids between start and end
with open(enc_file) as f: a=f.readlines()
for i in a:
    b=i.strip().split()
    if((float(start)<=float(b[0]) and float(end)>=float(b[0]))and(float(start)<=float(b[1]) and float(end)>=float(b[1]))):
        binlist.append(b[2])
print ("binlist:",binlist)

#calculate number of records between start and end
recsbtstartend=0
for i in records:
    if float(i[dim])>=float(start) and float(i[dim])<=float(end):
        recsbtstartend+=1
print ("records between start and end:",recsbtstartend)

with open(transformed_data) as f2: c=f2.readlines()

#calculate number of records in transformed data
recinbins=0
for i in c:
    d=i.strip().split()
    if d[3] in binlist:
        recinbins+=1
print ("records in transformed data:",recinbins)


#find the precision
precision=float(recsbtstartend)/float(recinbins)
print("Precision:", format(precision,'.3f'))
```

Output:



```
meena@meena-VirtualBox:~/Desktop/Project4$ python crypto-index-query.py breast-cancer_scale output_data_breast_cancer bin_encoding_file 2 -1.0 -0.3
python crypto-index-transdata.py input_data dimenion_i bin_width bin_encoding_file output_data
('binlist:', ['5708665', '4406947', '7229677'])
('records between start and end:', 508)
('records in transformed data:', 470)
('Precision:', '1.081')
```

Question 2.2 Develop a simple script that processes a number n, e.g., n=10, of randomly generated ranges for your queries. Note these ranges should be within the domain of dimension_i. Compute the average query result precision of the n ranges, for different bin_width settings: 0.02, 0.04, 0.06, 0.08, 0.1. You should try your script on the three datasets. Report the result precision and standard deviation (e.g., 0.82+/-0.05) in the following table, and also post your script here.

Program:

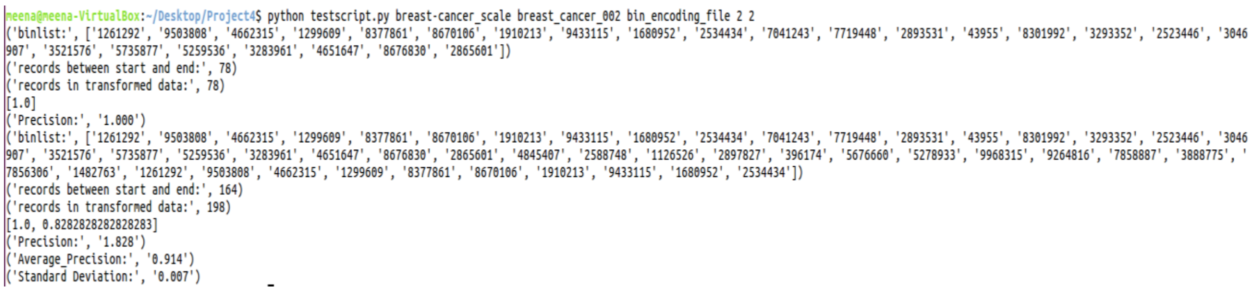
```
import os, sys, random,math
from readsvm import readSVM


def usage():
    print "python crypto-index-transdata.py input_data dimenion_i bin_width bin_encoding_file output_data"

if len(sys.argv)!=6:
    usage()

input_data = sys.argv[1]
dim = int(sys.argv[4])
enc_file = sys.argv[3]
transformed_data = sys.argv[2]
n=int(sys.argv[5])

target, records = readSVM(input_data)
binlist=[]
total_precision=0
precision=[]
diff=0
#iterate over n
for i in range(0,n):
    #randomly generating range
    start=random.uniform(-1.0,0.0)
    end=random.uniform(0.0,1.0)
    #add binids between start and end
    with open(enc_file) as f: a=f.readlines()
    for i in a:
        b=i.strip().split()
        if((float(start)<=float(b[0]) and float(end)>=float(b[0]))and(float(start)<=float(b[1]) and
float(end)>=float(b[1]))):
            binlist.append(b[2])
    print ("binlist:",binlist)

#calculate number of records between start and end
    recsbtstartend=0
    for i in records:
        if float(i[dim])>=float(start) and float(i[dim])<=float(end):
            recsbtstartend+=1
    print ("records between start and end:",recsbtstartend)

    with open(transformed_data) as f2: c=f2.readlines()

    #calculate number of records in transformed data
    recinbins=0
    for i in c:
        d=i.strip().split()
        if d[3] in binlist:
            recinbins+=1
    print ("records in transformed data:",recinbins)

    #find the precision
    total_precision+=float(recsbtstartend)/float(recinbins)
    precision.append(float(recsbtstartend)/float(recinbins))
    print precision
    print("Precision:", format(total_precision,'.3f'))
```

```
#print average precision
average_precision=float(total_precision)/n
print("Average_Precision:", format(average_precision,'.3f'))
#print standard deviation
for i in precision:
    diff+=math.pow((i-average_precision),2)
standard_deviation=float(diff)/n
print("Standard Deviation:", format(standard_deviation,'.3f'))
```

Output:

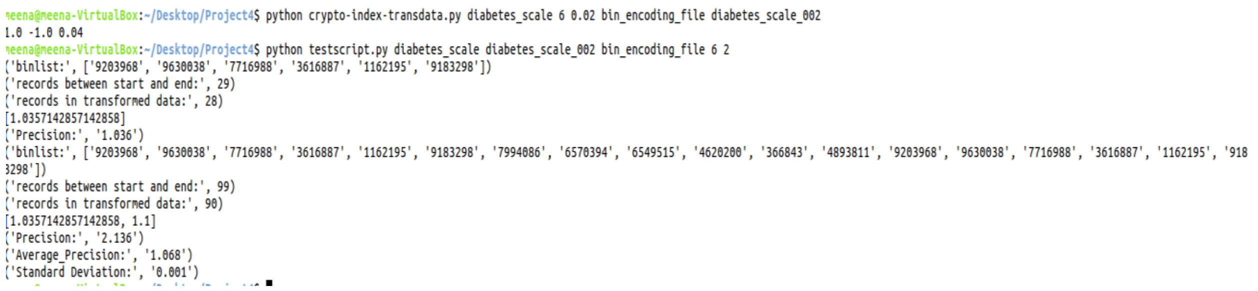
bin_width	***0.02***	***0.04***	***0.06***	***0.08***	***0.1***
breast_cancer	0.914 ± 0.007	0.704 ± 0.088	0.976 ± 0.112	1.125 ± 0.016	0.866 ± 0.018
diabetes	1.068 ± 0.001	0.881 ± 0.044	1.079 ± 0.240	1.945 ± 0.082	1.584 ± 0.489
heart	1.072 ± 0.003	1.024 ± 0.000	1.564 ± 0.236	2.918 ± 0.261	0.831 ± 0.045

Breast_cancer:



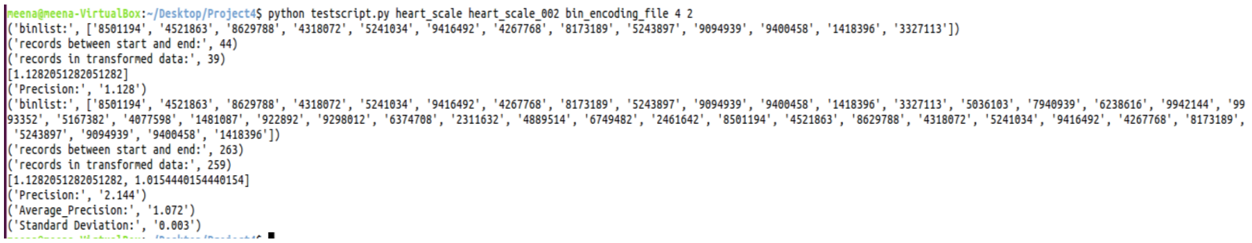
```
meena@meena-VirtualBox:~/Desktop/Project4$ python testscript.py breast-cancer_scale breast_cancer_002 bin_encoding_file 2 2
('binlist:', ['1261292', '9503808', '4662315', '1299609', '8377861', '8670106', '1910213', '9433115', '1680952', '2534434', '7041243', '7719448', '2893531', '43955', '8301992', '3293352', '2523446', '3046
907', '3521576', '5735877', '5259536', '3283961', '4651647', '8676830', '2865601'])
('records between start and end:', 78)
('records in transformed data:', 78)
[1.0]
('Precision:', '1.000')
('binlist:', ['1261292', '9503808', '4662315', '1299609', '8377861', '8670106', '1910213', '9433115', '1680952', '2534434', '7041243', '7719448', '2893531', '43955', '8301992', '3293352', '2523446', '3046
907', '3521576', '5735877', '5259536', '3283961', '4651647', '8676830', '2865601', '4845407', '2588748', '1126526', '2897827', '396174', '5676660', '5278933', '9968315', '9264816', '7858887', '3888775', '
7856306', '1482763', '1261292', '9503808', '4662315', '1299609', '8377861', '8670106', '1910213', '9433115', '1680952', '2534434'])
('records between start and end:', 164)
('records in transformed data:', 198)
[1.0, 0.8282828282828283]
('Precision:', '1.828')
('Average_Precision:', '0.914')
('Standard Deviation:', '0.007')
```

Diabetes:



```
meena@meena-VirtualBox:~/Desktop/Project4$ python crypto-index-transdata.py diabetes_scale 6 0.02 bin_encoding_file diabetes_scale_002
1.0 -1.0 0.04
meena@meena-VirtualBox:~/Desktop/Project4$ python testscript.py diabetes_scale diabetes_scale_002 bin_encoding_file 6 2
('binlist:', ['9203968', '9630038', '7716988', '3616887', '1162195', '9183298'])
('records between start and end:', 29)
('records in transformed data:', 28)
[1.0357142857142858]
('Precision:', '1.036')
('binlist:', ['9203968', '9630038', '7716988', '3616887', '1162195', '9183298', '7994086', '6570394', '6549515', '4620200', '366843', '4893811', '9203968', '9630038', '7716988', '3616887', '1162195', '918
3298'])
('records between start and end:', 99)
('records in transformed data:', 90)
[1.0357142857142858, 1.1]
('Precision:', '2.136')
('Average_Precision:', '1.068')
('Standard Deviation:', '0.001')
```

Heart:



```
meena@meena-VirtualBox:~/Desktop/Project4$ python testscript.py heart_scale heart_scale_002 bin_encoding_file 4 2
('binlist:', ['8501194', '4521863', '8629788', '4318072', '5241034', '9416492', '4267768', '8173189', '5243897', '9094939', '9400458', '1418396', '3327113'])
('records between start and end:', 44)
('records in transformed data:', 39)
[1.1282051282051282]
('Precision:', '1.128')
('binlist:', ['8501194', '4521863', '8629788', '4318072', '5241034', '9416492', '4267768', '8173189', '5243897', '9094939', '9400458', '1418396', '3327113', '5036103', '7940939', '6238616', '9942144', '99
93352', '5167382', '4077598', '1481087', '922892', '9298012', '6374708', '2311632', '4889514', '6749482', '2461642', '8501194', '4521863', '8629788', '4318072', '5241034', '9416492', '4267768', '8173189',
 '5243897', '9094939', '9400458', '1418396'])
('records between start and end:', 263)
('records in transformed data:', 259)
[1.1282051282051282, 1.0154440154440154]
('Precision:', '2.144')
('Average_Precision:', '1.072')
('Standard Deviation:', '0.003')
```

Question 2.3 Discuss the potential attacks to the crypto-index method.

Background knowledge attack:
In Crypto-index method data is divided and stored in different bins to ensure privacy and security. If the data is stored without anonymizing it, for example, if it contains some personal identifying information, then it could be easy for the attackers to link the known data and extract the information. Therefore, it is essential to remove any personal data.
Man in the middle attack:
It is a type of attack where secret malicious actor is introduced between the client and the server communication and gain access to information that was transferred between the two parties.