ECE 194B Final Project

Heart Disease Detector

Shuhe Jiang (shuhe@ucsb.edu), Kuangyi Mi (kmi00@ucsb.edu)

Introduction

Cardiovascular diseases (Heart Diseases or CVDS) are the number one cause of death global. The disease takes about 17.9 million lives every year, consisting of about 31% global death yearly. For every 37 seconds, one person dies in the United States from CVDS and about 647,000 Americans die from heart disease each year, which is one in every four death. At the same time, heart disease costs the United States about $219 billion each year. The astonishing data stimulate our interests to investigate further into CVDS. Though we are not capable of curing and saving people from CVDS, but we want to at least set up an optimization problem that can help predict the potential threaten of CVDS to people according to several features that the patient shows. We hope those with high risks can take special care of themselves to prevent the tragedy happens.

Since the primary data requires various tests and lab analysis and are not easily accessible. After doing some research, the data from UCI Machine Learning Data sets are used. In total, there are 303 instances in the data set and 76 attributes. However, only 14 attributes are actually matter in this lab (so, we eliminated many unrelated attributes). The goal of this project is to predict weather a patient gets the CVDS or not by giving in total 14 attributes. In the original data set, the presence of heart disease in the patient is valued from 0 (no presence) to 4 (the most series disease). However, due to the limitation of the model that we build, we adjust the data. In our result, the patient with heart disease are marked as 1 and those without presence of heart disease are marked as -1.

This report will explain the mathematical model used, including the assumptions made, explanation of decision variables, constraint and the objective function. Then the attempt to solve the problem with different methods. To make a good prediction, we attempted both the logistic regression and kernel to build the prediction. Then the accuracy of prediction are compared to decide the better solution.

Mathematical model

In the process of constructing our model of heart disease predictor, we tried and compared different mathematical models and then decided our final model.

In Stage 1, we firstly decided to apply the support vector machine (SVM) with linear kernel to our heart disease dataset in order to successfully find the proper classifier ω with a support constant b.

General problem preparation:

1. For the provided dataset, we first decided to shuffle the data to have a more random dataset.

2. We divided our data into 2 categories: One of them contains all the potential influential factors on heart disease, and the second one is composed of the indicators that specify whether our participants have heart disease or not.
3. In Step 3, since we plan to use the binary classification, we reassign values to all the indicators thus making them either 1 or -1. Also, in order to have our code work better, we preprocessed our features, so they will be inside the same range.
4. Since we cannot directly determine what gamma (the degree of tolerance of misclassified datapoint) will work best for our model, we constructed a vector containing different gammas and let the computer make the decision for us.

The SVM model setting:

1.The basic idea behind the SVM is that we want to construct a proper hyperplane that helps us distinguish the diagonalized patients and healthy participants by putting them either above the hyperplane or below it. In order to achieve this goal, we need to first identify the equation of a hyperplane: $\omega^T x + b = 0$

2.Knowing the equation of the hyperplane, we have had our first 2 decision variables: ω and b. At the same time, we also need to assign them with the indicators as shown below:

$$\text{If } y = \text{-}\ 1 \Rightarrow \omega^T x + b < 0;$$
$$\text{If y} = \ \ 1 \Rightarrow \omega^T x + b > 0;$$

3. By observing the formula above, we know that if we multiply the indicators with their corresponding product of the ideal classifier and our features related to heart disease, we will always have positive values since these 2 numbers will be both positive or both negative. Consequently, the formula is now shown as below:

$$y_i \times \left(\omega^T x_i + b\right) > 0$$

4.From week 5 lecutures, to make this hyperplane greater or smaller than 0 is the same as greater than 1 and smaller than -1

(Explanation of why we reassign values of heart disease indicators with 1 and -1). By adding another supporting vector decision variable eta, we approximately finish our constraints for the optimization problem.

$$y_i \times \left(\omega^T x_i + b\right) > 1 \text{-} \eta;$$
$$\eta \geq 0;$$

5.In order to solve for the classifier, or find this proper hyperplane, only knowing the constraints above won't be enough. The original idea for the is to maximize the "geometric margin." However, in order to accomplish this optimization problem into a convex one, we change it to minimize the 2-norm of the classifier coeffect—ω. Here, since this real-life data might not be perfectly linearly separable, we introduced another constant gamma which denotes the degree of tolerance of misclassified datapoints. As a result, the whole SVM problem setup is shown below, and our target classifier will be solved by transferring the formula below into MATLAB language:

Objective Function: $\min_{\omega,\eta,b} \|\omega\|_2 + \|\eta\|_1 \times \gamma$

Constraints: $s.t.\ y_i \times \left(\omega^T x_i + b\right) > 1 - \eta;$
$\eta \geq 0;$

In Stage 2, since we have the wish to further improve our classification, we decided to use the logistic regression to predict whether people with certain features have heart disease or not. In order to realize the logistic regression

better, we conducted similar problem preparations as what we did for Stage 1.

The Logistic Regression (with gradient descent) Model Setting:

1.Gradient descent is a method of finding the minimum optimal value for a convex problem by moving towards a “lower” direction step by step. Accoring to the lecture, we know that a new ω—the classifier—is equal to the old ω plus the gradient of in-sample error multiplied by the given step size alpha.

$$-\frac{1}{N}\times\sum_{i=1}^{N}\frac{y_i\times x_i}{1+e^{y_i\times\omega^T x_i}}$$

2.After finding ω, we want to test it using the theta function which is shown below and designed to tell us probability. However, since instead of using them to predict the probability of participants in test set having heart disease, we want to use them to classify whether they have the disease or not, we divide the probability into 2 categories using the boundary of 50%.

$$\theta(\omega^T x)=\frac{1}{1+e^{-\omega^T x}}$$

Solution

The following is out attempt to solve this optimization problem through ***Linear Kernel***

```
%Final Project Linear Kernel (SVM) Solution

load('processed.mat')
r=10;%try 5 different gamma value
x=processed(:,1:13);
y=processed(:,14);
%divide this dataset into attributes and result
for i=1:297
    if y(i)==0
        y(i)=y(i)-1;
    else
        y(i)=1;
    end
end
%make no presence of heart disease=-1&presence of heart disease=1
shuffle = [x,y];
shuffle = shuffle(randperm(size(shuffle,1)), :);
x = shuffle(:,1:size(x,2));
y = shuffle(:,end);
clear shuffle
%shuffle the data
m=158;
x_train=x(1:m,:);
x_test=x(m+1:297,:);

y_train=y(1:m,:);
y_test=y(m+1:297,:);
%divide both x and y into training(size=163) and test sets(size=134)
t1=[];
t2=[];

cvx_begin
```

```
    variables a(13) b eta(m)
    minimize polyval(norm(a,2),1)+norm(eta,1)*r
    subject to
    y_train.*(x_train*a-b)>=1-eta;
    eta>=0;
cvx_end
```

```
 Calling SDPT3 4.0: 647 variables, 316 equality constraints
 ------------------------------------------------------------

  num. of constraints = 316
  dim. of socp   var  = 330,   num. of socp blk  = 159
  dim. of linear var  = 316
  dim. of free   var  =  1 *** convert ublk to lblk
 *****************************************************************
    SDPT3: Infeasible path-following algorithms
 *****************************************************************
  version  predcorr  gam  expon  scale_data
     NT      1      0.000   1        0
 it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
 -------------------------------------------------------------------
  0|0.000|0.000|2.4e+01|4.6e+01|2.1e+05| 2.238199e+03  0.000000e+00| 0:0:00| chol  1  1
  1|0.465|0.241|1.3e+01|3.5e+01|1.6e+05| 2.461002e+03  5.828458e+02| 0:0:00| chol  1  1
  2|0.341|0.240|8.4e+00|2.7e+01|1.3e+05| 2.724849e+03  2.028059e+03| 0:0:00| chol  1  1
  3|0.361|0.162|5.4e+00|2.2e+01|1.2e+05| 3.257156e+03  2.440242e+03| 0:0:00| chol  1  1
  4|0.659|0.459|1.8e+00|1.2e+01|7.5e+04| 4.341742e+03  2.756379e+03| 0:0:00| chol  1  1
  5|1.000|0.676|2.3e-05|3.9e+00|2.9e+04| 5.257257e+03  1.165551e+03| 0:0:00| chol  1  1
  6|1.000|0.969|2.8e-06|1.2e-01|3.9e+03| 3.610849e+03  2.932092e+02| 0:0:00| chol  1  1
  7|0.968|0.375|6.2e-08|7.6e-02|9.8e+02| 1.163679e+03  3.004828e+02| 0:0:01| chol  2  2
  8|0.896|0.428|4.0e-05|4.4e-02|5.1e+02| 8.199931e+02  3.505298e+02| 0:0:01| chol  2  2
  9|1.000|0.296|5.5e-06|3.1e-02|3.2e+02| 6.727563e+02  3.758465e+02| 0:0:01| chol  2  2
 10|0.878|0.446|2.8e-06|1.7e-02|2.0e+02| 5.966846e+02  4.101055e+02| 0:0:01| chol  2  2
 11|1.000|0.339|5.2e-07|1.1e-02|1.1e+02| 5.292200e+02  4.254877e+02| 0:0:01| chol  2  2
 12|1.000|0.398|1.7e-07|6.8e-03|7.1e+01| 5.065412e+02  4.396463e+02| 0:0:01| chol  2  2
 13|0.918|0.387|4.8e-08|4.1e-03|4.3e+01| 4.897596e+02  4.496953e+02| 0:0:01| chol  2  2
 14|1.000|0.471|1.6e-08|2.2e-03|2.1e+01| 4.790271e+02  4.590493e+02| 0:0:01| chol  1  2
 15|1.000|0.443|3.4e-09|1.2e-03|1.1e+01| 4.747212e+02  4.644644e+02| 0:0:01| chol  2  2
 16|1.000|0.678|1.5e-09|3.9e-04|3.4e+00| 4.728212e+02  4.696557e+02| 0:0:01| chol  1  1
 17|0.978|0.909|1.1e-09|3.6e-05|2.9e-01| 4.723625e+02  4.720938e+02| 0:0:01| chol  1  1
 18|0.987|0.987|6.5e-11|4.7e-07|3.8e-03| 4.723431e+02  4.723396e+02| 0:0:01| chol  1  1
 19|0.989|0.989|1.1e-10|2.4e-08|4.5e-05| 4.723428e+02  4.723428e+02| 0:0:01| chol  1  1
 20|0.596|0.944|4.6e-11|4.2e-10|4.5e-06| 4.723428e+02  4.723428e+02| 0:0:01|
   stop: max(relative gap, infeasibilities) < 1.49e-08
 -------------------------------------------------------------------

 number of iterations   = 20
 primal objective value =  4.72342813e+02
 dual   objective value =  4.72342809e+02
 gap := trace(XZ)       = 4.50e-06
 relative gap           = 4.76e-09
 actual relative gap    = 4.52e-09
 rel. primal infeas (scaled problem)   = 4.62e-11
 rel. dual     "        "       "      = 4.20e-10
 rel. primal infeas (unscaled problem) = 0.00e+00
 rel. dual     "        "       "      = 0.00e+00
 norm(X), norm(y), norm(Z) = 2.4e+01, 9.8e+01, 1.9e+02
 norm(A), norm(b), norm(C) = 4.1e+03, 1.4e+01, 1.3e+02
 Total CPU time (secs)  = 0.96
```

```
 CPU time per iteration = 0.05
 termination code       =  0
 DIMACS: 3.1e-10  0.0e+00  4.8e-09  0.0e+00  4.5e-09  4.8e-09
-------------------------------------------------------------------


-------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +472.343
```

```
TP=0;%stands for true positive
FP=0;%stands for false positive
FN=0;%stands for false negative
TN=0;%stands for true negative

tp=[];%create vectors for true positive
fp=[];%create vectors for false positive
tn=[];%create vectors for false negative
fn=[];%create vectors for true negative


for j=1:size(x_test,1)
   new=y_test(j)*(a'*x_test(j,:)'-b);
    if new > 0
        if (y_test(j)==1)
            TP=TP+1;
            tp(TP)=new;
        else
            TN=TN+1;
            tn(TN)=new;
        end
    else
        if (y_test(j)==-1)
            FP=FP+1;
            fp(FP)=new;
        else
            FN=FN+1;
            fn(FN)=new;
        end
    end
end %used to calculate the number of true positive & true negative

    correct_rate=(TN+TP)/size(x_test,1);
    disp(['the accuracy rate is: ',num2str(correct_rate)]);
```

```
 the accuracy rate is: 0.80576
```

Comment:

In the coding process, we tried gama to be 0.01,0.1,0.5,1,10. We found when gama=10, the accuracy is highest. However, when we continue make gama larger, nothing changes. So we decided to set gama=10 here.

The following is out attempt to solve this optimization problem through ***Logistic Regression***

```
%Final Project Logistic Regression Solution
load("processed.mat")
%import our data set

x=processed(:,1:13);
y=processed(:,14);
%define x and y

for i=1:size(x,1)
    if y(i)==0
        y(i)=y(i)-1;
    else
        y(i)=1;
    end
end
%make y into 2 categories(1:having heart disease 0:no presence of heart disease)

shuffle = [x,y];
shuffle = shuffle(randperm(size(shuffle,1)), :);
x = shuffle(:,1:size(x,2));
y = shuffle(:,end);
clear shuffle
%shuffle the data for train sets

m=178;
x_train=x(1:m,:);
x_test=x(m+1:297,:);

y_train=y(1:m,:);
y_test=y(m+1:297,:);
%divide both x and y into training and test sets

x_train= log(x_train+0.1);
x_test= log(x_test+0.1);
%rescale our data

w=zeros(13,1); %prepared weight vector
a=0.01;%step size

iteration=100000;


for i = 1:iteration
   grad=-sum(y_train.*x_train./(1+exp(y_train.*(x_train*w))))./size(x_train,1);
   %use the sum function to calculate the gradient of in sample error
   %the sum function will return us a 1x13 matrix
   w=w+a*(-grad'); %final weight
end

TP=0;%stands for true positive
FP=0;%stands for false positive
FN=0;%stands for false negative
TN=0;%stands for true negative
```

```
prob_mix=zeros(147,1);
tp=[];%create vectors for true positive
fp=[];%create vectors for false positive
tn=[];%create vectors for false negative
fn=[];%create vectors for true negative
for j=1:size(x_test,1)
    prob=1./(1+exp(-x_test(j,:)*w));
    prob_mix(j,1) = prob;
    if prob > 0.5
        if (y_test(j)==1)
            TP=TP+1;
            tp(TP)=prob;
        else
            FP=FP+1;
            fp(FP)=prob;
        end
    else
        if (y_test(j)==-1)
            TN=TN+1;
            tn(TN)=prob;
        else
            FN=FN+1;
            fn(FN)=prob;
        end
    end
end %used to calculate the number of true positive & true negative

    correct_rate=(TN+TP)/size(x_test,1);
    disp(['the accuracy rate is:',num2str(correct_rate)]);
```

```
the accuracy rate is:0.86555
```

Results and discussion

We had sevveral attempt to improve the accuracy of the code. The following are the presentation of the accuracy and the real situation of classfication:

- ***Linear Kernel***

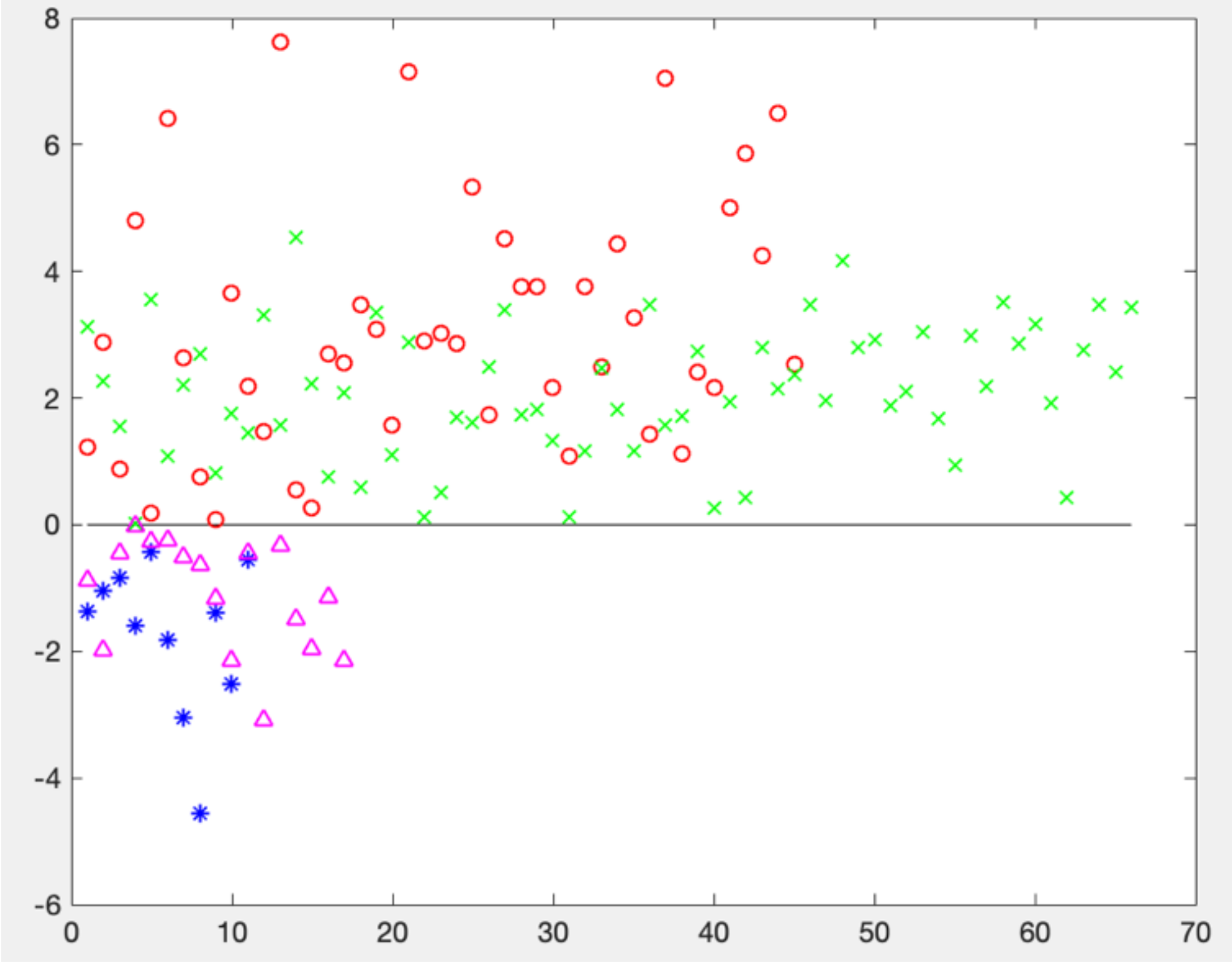
  The accuracy rate is: 0.80576(The rate is subjected to change, but it floats around 0.80

  The following are the presentation of the result:

```
 %Graph to show the distribution of data and classification
xc=1:1:length(tp);
xd=1:1:size(tn,2);
xe=1:1:size(fp,2);
xf=1:1:size(fn,2);
new1=max(length(xc),length(xd));
new2=max(length(xe),length(xf));
new=max(new1,new2);

plot([1 new],[0 0],'k-')
hold on
```

```
  scatter(xc,tp,'ro')
  hold on
  scatter(xd,tn,'gx')
  hold on
  scatter(xe,fp,'b*')
  hold on
  plot(xf,fn,'m^')
  hold on
```



Red 'o' represents 'true positive'

Green 'x' represents 'true negative'

Blue '*' represents 'false postive'

Pink 'Δ' represents 'false negative'

In the graph above, all the data that are "true positive" and "true negative" are above the line y=0. These points are the data that are predicted accurately and those points under 0 are the data that should be positive but predicted negative, or should be negative but predicted positive.

In the graph above, most of the data are predicted right, which means most of those who presents heart disease

can be distinguished and can get necessary help in time.

However, there are still some parents who have disease and not be distinguished, which is the limitation of our model.

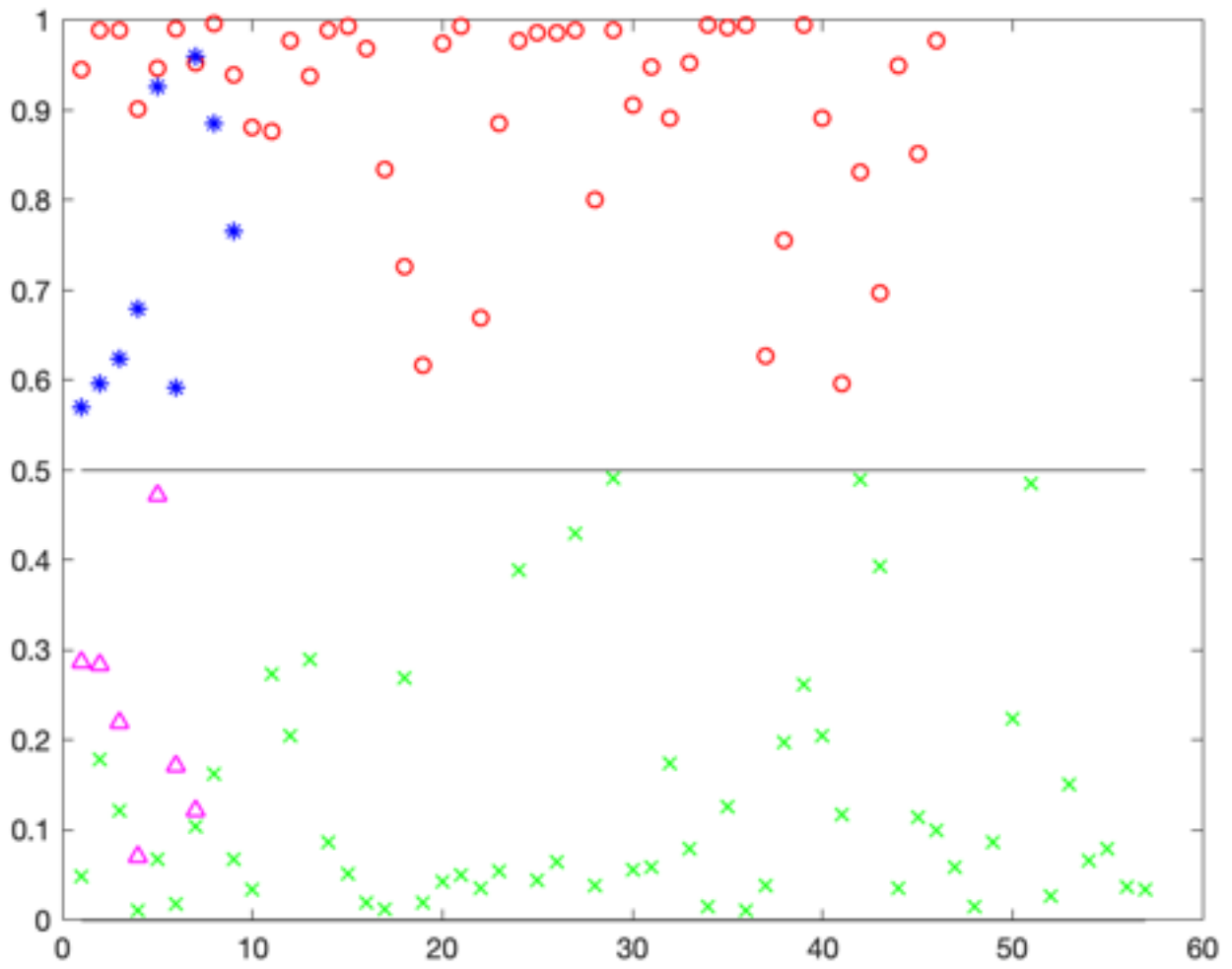
- ***Logistic Regression***

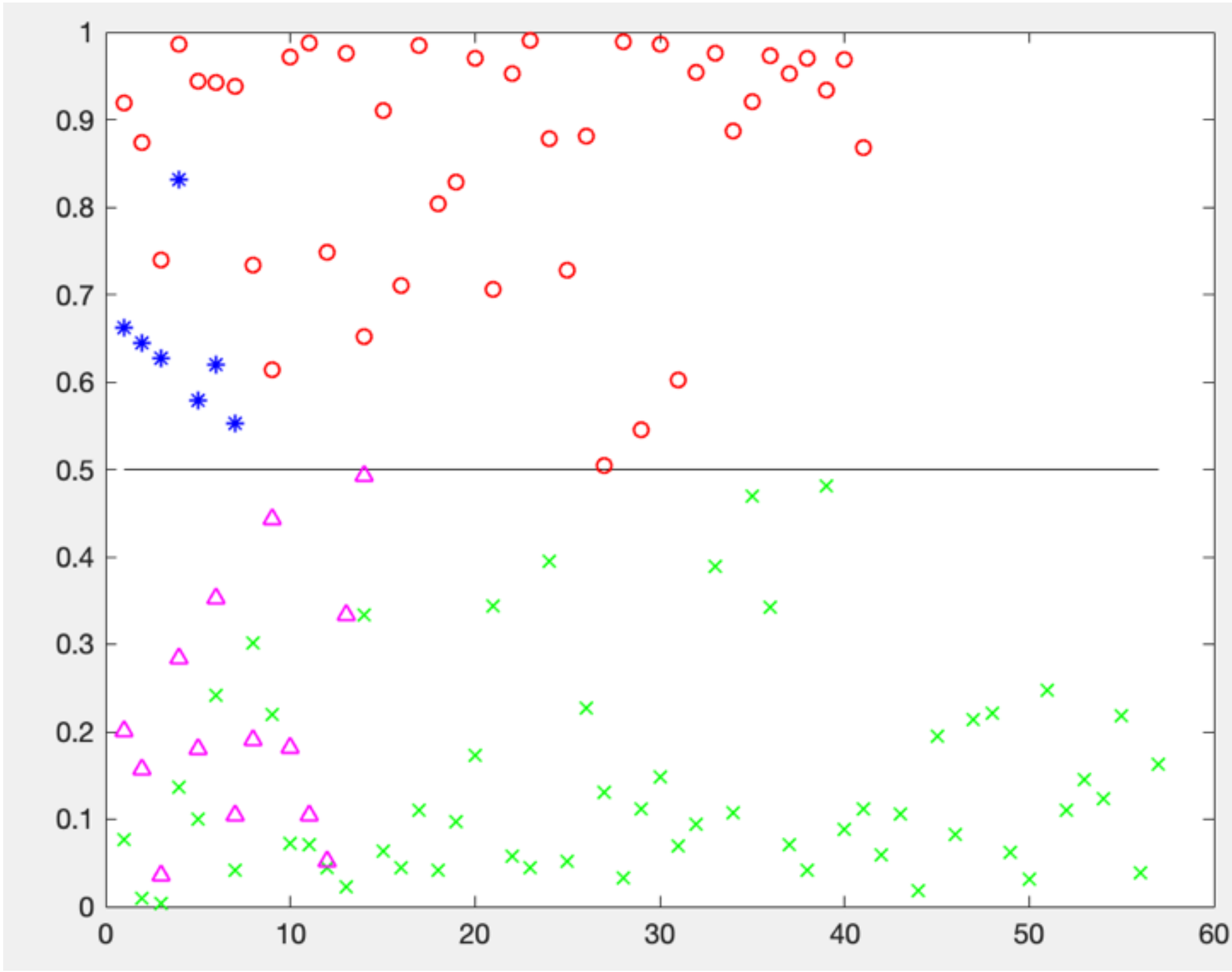
The accuracy rate is: 0.86555(The rate is subjected to change, but it floats around 0.80)

The following are the presentation of the result:

```
 %Graph to show the distribution of data and classification
xm=1:1:length(tp);
xn=1:1:size(tn,2);
xp=1:1:size(fp,2);
xq=1:1:size(fn,2);
new1=max(length(xm),length(xn));
new2=max(length(xp),length(xq));
new=max(new1,new2);

plot([1 new],[0.5 0.5],'k-')
hold on
scatter(xm,tp,'ro')
hold on
scatter(xn,tn,'gx')
hold on
scatter(xp,fp,'b*')
hold on
plot(xq,fn,'m^')
```





Red 'o' represents 'true positive'

Green 'x' represents 'true negative'

Blue '*' represents 'false postive'

Pink 'Δ' represents 'false negative'

In the graph above, the data that are “true positive” and those data that are “false positive” are above the line y=0.5, in which those blue ‘*’ are patients who do not have disease but be distinguished as diseased in our model. The data that are “true negative” and those “false negative” are below the line y=0.5, in which those pink points are patients who have disease but not be detected.

Comparision of Two Models

The Linear Regression and Logistic Regress have both advantages and disadvantages.

In generally, ***Linear Kernal*** is limited to linear relationships. By its nature, linear regression only looks at linear relationships between dependent and independent variables.

Linear regression focus on the relationship between the mean of the dependent variable and the independent variables. At the same time, Linear Regression is sensitive to outliers and assume that the data are independent, which means that the scores of one subject have no relation with those of another. This creates a hard limitation in clustering applications where variables need to be clustered based on space and time.

Logistic Regression is simple, efficient, easy interpretation and use limited computational resources and it allows easy regularization of outputs to prevent overfitting, yielding probabilities as prediction results. Logistic Regression allows easy *model updating* using *stochastic gradient descent*. Logistic Regression models does not get effected to predict output probabilities on removal of variables uncorrelated to the output or multi-collinear variables.

Logistic regression's greatest disadvantage is fails to solve non-linear problems and it underperforms when there are multiple or non-linear decision boundaries. It fails to capture more complex relationships. Logistic Regression can only predict a categorical outcome with discrete probability outcome.

In this case, our results derived from both ***Linear Kernal*** and ***Logistic Regression*** since our data basically fit the linear relationship(the data set that we). So, both cases all have around 80% accuracy.

Conclusion

In our attempt, we solved the optimization problem by training 60% of the data set and test the rest. In our result, we can classify and predict weather a patient presents the heart disease by their 14 attributes according to their personal healthy situation.

In our original plan, we tried to classify the data to 0,1,2,3,4 (0 means no presence of, 0-4 means more serious disease). Therefore, we can build the system to help doctors and patient to help them find the potential patients.

However, since the accuracy of classified the data into two category in only 80%. To make sure the accuracy rate to classify the data into 5, the accuracy of two classification must be higher than 95%.

Since those who does not have disease but tested positive can be exclude through further examination. However, those who has disease but not detected are quite dangerous. Therefore, since such kind of predictor should be quite accurate to save people's life.

To build a more accurate heart disease predictor, more attribute should be include and more past disease data should be collected to train the model and a different model may be applied (maybe the non-linear kernel) to classify the data in a different way.

Appendix

Attribute Information

1.Age

2.Sex

3.Chest pain type

- Value 1: Typical Angina
- Value 2: Atypical Angina
- Value 3: Non-anginal Pain

- Value 4: Asymptomatic

4.TRESTBPS: Resting Blood Pressure- Measure in mmHG on admission to hospital

5.CHOL: serum cholestoral in mg/dl

6.FBS: (fasting blood sugar > 120 mg/dl)

- Value 1: True
- Value 0: False

7.RESTECG: resting electrocardiographic results

- Value 0: normal
- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

8.THALACH: maximum heart rate achieved

9.EXANG: exercise induced angina (1 = yes; 0 = no)

- Value 1: Yes
- Value 0: NO

10.Old peak = ST depression induced by exercise relative to rest

11.slope: the slope of the peak exercise ST segment

- Value 1: up-sloping
- Value 2: flat
- Value 3: down-sloping

12.CA: number of major vessels (0-3) colored by flourosopy

13.THAL

- Value 3: Normal
- Value 6: Fixed defect
- Value 7: Reversible Defect

14.NUM: diagnosis of heart disease (angiographic disease status)

1. Value 0: < 50% diameter narrowing
2. Value 1: > 50% diameter narrowing

Reference

https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1

https://archive.ics.uci.edu/ml/datasets/Heart+Disease

https://towardsdatascience.com/a-comprehensive-study-of-linear-vs-logistic-regression-to-refresh-the-basics-7e526c1d3ebe