% Psych 216A: Statistics and data analysis in MATLAB
% Homework 8 (Classification models)

% 1. For the following set of data, fit a logistic regression model using
% an input space that is expanded to include quadratic terms (i.e. $x_1^2,$
% $x_2^2$, $x_1x_2$). Visualize the data and the decision boundary associated with
% the fit of the model (this fit of the model should use all of the data).
% Then, use leave-one-out cross-validation to obtain unbiased estimates of
% model accuracy. Report the accuracy level (i.e. classification performance
% expressed in terms of percent correct) in the figure title.

x1 = [1.4 2.8 1.1 -0.4 0.91 -0.023 -0.37 1.7 0.61 0.69 1.4 -0.7 1.2 -1.6 ...
    2.4 2.5 1.4 0.35 1.3 1.8 2.3 1.4 0.53 1.4 -1.3 1.1 0.17 0.18 1.1 ...
    0.45 0.12 0.81 3 1.5 1 -0.91 0.5 1.1 0.92 0.13 1 -0.22 2.1 -0.27 ...
    -1.1 0.34 -1.1 1.3 0.62 -1.1];
x2 = [0.17 0.94 0.44 -0.36 -0.39 -0.2 0.82 -0.29 0.69 -1.3 -0.91 0.99 0.72 ...
    -0.99 -1.4 -0.87 -0.0019 -0.6 0.89 -0.025 -1.2 -0.73 0.079 0.13 -1.2 ...
    -1.6 -0.88 -0.92 0.53 -1.9 0.54 -0.66 -1 -2 1.9 2.4 -0.89 0.17 -0.89 ...
    0.14 2.5 -0.2 1 -1.2 -0.47 -1 0.4 -1 0.33 -1.9];
y = [1 1 1 1 1 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 0 0 1 1 0 ... 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0]' ;

% define optimization options
options = optimset('Display','off','FunValCheck','on','MaxFunEvals',Inf, ... 'MaxIter',Inf,'TolFun',1e-4,'TolX',1e-4);

% define logistic function (this should take a matrix of values and
% apply the logistic function to each value)
logfun = @(x) 1./(1+exp(-x));

% construct regressor matrix (data points x regressors)
X = [x1 x2 x1.^2 x2.^2 x1.*x2];
X(:,end+1) = 1; % need a constant regressor too

% define initial seed
seed = zeros(1,size(X,2));

% the cost function to minimize is the negative-log-likelihood.
% here, the eps is a hack to ensure that the log returns a finite number
costfun = @(pp) -sum((y .* log(feval(logfun,X*pp')+eps)) + ... ((1-y) .* log(1-feval(logfun,X*pp')+eps)));

% fit model to all of the data
params = lsqnonlin(costfun,seed,[],[],options);

% perform leave-one-out cross-validation
prediction = zeros(size(y));
for p=1:length(y)
    % define training and testing indices
testix = p;
trainix = setdiff(1:length(y),testix);

% estimate the parameters of the model
costfun = @(pp) -sum((y(trainix) .* ...
    log(feval(logfun,X(trainix,:)*pp')+eps)) + ...
    ((1-y(trainix)) .* ...
    log(1-feval(logfun,X(trainix,:)*pp')+eps)));
paramsB = lsqnonlin(costfun,seed,[],[],options);

% compute the prediction of the model
prediction(testix) = (X(testix,:)*paramsB' >= 0.5);

end

% calculate percent correct
pct = sum(prediction==y) / length(y) * 100;

% visualize
figure; hold on;
colormap(gray);
    % visualize the data points
h1 = scatter(x1,x2,50,y,'filled');
set(h1,'MarkerEdgeColor','k');
    % prepare a grid of points to evaluate model at
ax = axis;
xvals = linspace(ax(1),ax(2),300);
yvals = linspace(ax(3),ax(4),300);
[xx,yy] = meshgrid(xvals,yvals);
    % construct regressor matrix
xx = xx(:);
yy = yy(:);
Xtemp = [xx yy xx.^2 yy.^2 xx.*yy];
Xtemp(:,end+1) = 1;
    % evaluate model at the points
predictionimage = reshape(Xtemp*params' >= 0.5,[length(yvals) length(xvals)]);
    % visualize the image using a large color range
    % so that the colors turn out to be shades of gray
h2 = imagesc(predictionimage,[-1 2]);
    % the default coordinate system is 1:N where N is the number of pixels
    % along each dimension. we have to move the image to the proper
    % coordinate system.
set(h2,'XData',xvals,'YData',yvals);
    % send the image to the bottom so that we can see the data points
uistack(h2,'bottom');
    % restore previous axis range
axis(ax);
    % finish up
xlabel('x_1');
ylabel('x_2');
title(sprintf('Cross-validated classification performance = %.1f%%',pct));