10.1

%This function takes in a data matrix X and a label

%vector y and outputs the average cat image and average dog image.

function [avgcat, avgdog] = average\_pet(X,y)

%FILL IN CODE

[D, C] = size(X);

count = 1;

for i = 1:D

if y(i) == -1

catM(count,:) = X(i,:);

count = count + 1;

end

end

count = 1;

for i = 1:D

if y(i) == 1

dogM(count,:) = X(i,:);

count = count + 1;

end

end

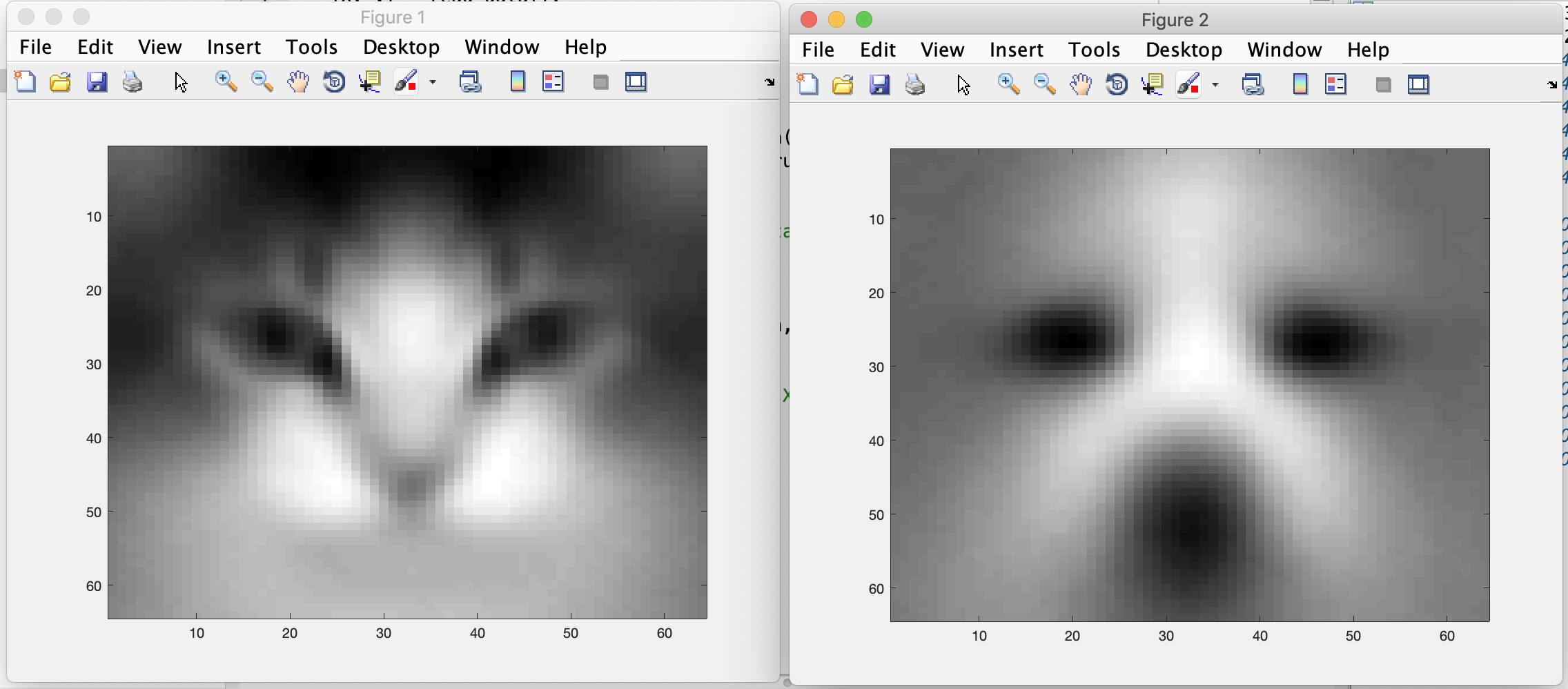
for j = 1:C

avgcat(j) = mean(catM(:,j));

avgdog(j) = mean(dogM(:,j));

end

end



10.2

%This function takes in a training data matrix Xtrain, training

%label vector ytrain and uses them to compute the average cat

%and dog vectors. It also takes in a data matrix Xrun and

%produces a vector of label guesses yguess, corresponding to whether

%each row of Xrun is closer to the average cat or average dog.

function yguess = closest\_average(Xtrain,ytrain,Xrun)

%FILL IN CODE

[DXtrain, CXtrain] = size(Xtrain);

count = 1;

for i = 1:DXtrain

if ytrain(i) == -1

catMtrain(count,:) = Xtrain(i,:);

count = count + 1;

end

end

count = 1;

for i = 1:DXtrain

if ytrain(i) == 1

dogMtrain(count,:) = Xtrain(i,:);

count = count + 1;

end

end

for j = 1:CXtrain

avgcattrain(j) = mean(catMtrain(:,j));

avgdogtrain(j) = mean(dogMtrain(:,j));

end

[DXrun, CXrun] = size(Xrun);

for i = 1:DXrun

rowdiffC(i,:) = Xrun(i,:) - avgcattrain;

rowdistC(i) = (sum((rowdiffC(i,:).^2))).^(1/2);

rowdiffD(i,:) = Xrun(i,:) - avgdogtrain;

rowdistD(i) = (sum((rowdiffD(i,:).^2))).^(1/2);

if rowdistC(i) > rowdistD(i)

yguess(i) = 1;

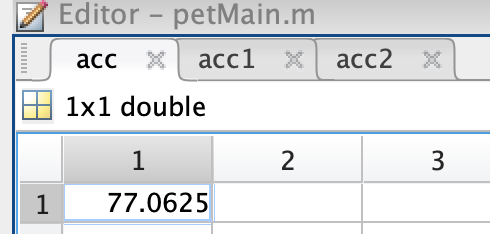
else

yguess(i) = -1;

end

end

end



10.3

%This function takes in a training data matrix Xtrain, training

%label vector ytrain and uses them to compute the average cat

%and dog vectors. It also takes in a data matrix Xrun and

%produces a vector of label guesses yguess. Each guess is found

%by searching through Xtrain to find the closest row, and then

%outputting its label.

function yguess = nearest\_neighbor(Xtrain,ytrain,Xrun)

%FILL IN CODE

[DXtrain, CXtrain] = size(Xtrain);

[DXrun, CXrun] = size(Xrun);

yguess = zeros(DXrun,1);

for i = 1:DXrun

nearest = Inf;

for j = 1:DXtrain

d = norm(Xrun(i,:) - Xtrain(j,:));

if d < nearest

nearest = d;

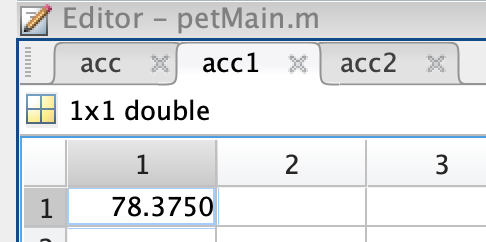
yguess(i) = ytrain(j);

end

end

end

end



10.4

%This function takes in a training data matrix Xtrain, training

%label vector ytrain and uses them to compute ordinary-least-squares

%vector b. It also takes in a data matrix Xrun and

%produces a vector of label guesses yguess, corresponding to the sign

%of the linear prediction.

function yguess = linear\_regression(Xtrain,ytrain,Xrun)

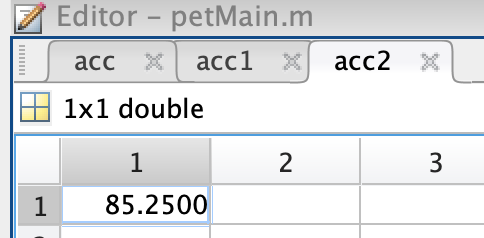
%FILL IN CODE

bols = pinv((Xtrain')\*Xtrain)\*(Xtrain')\*ytrain;

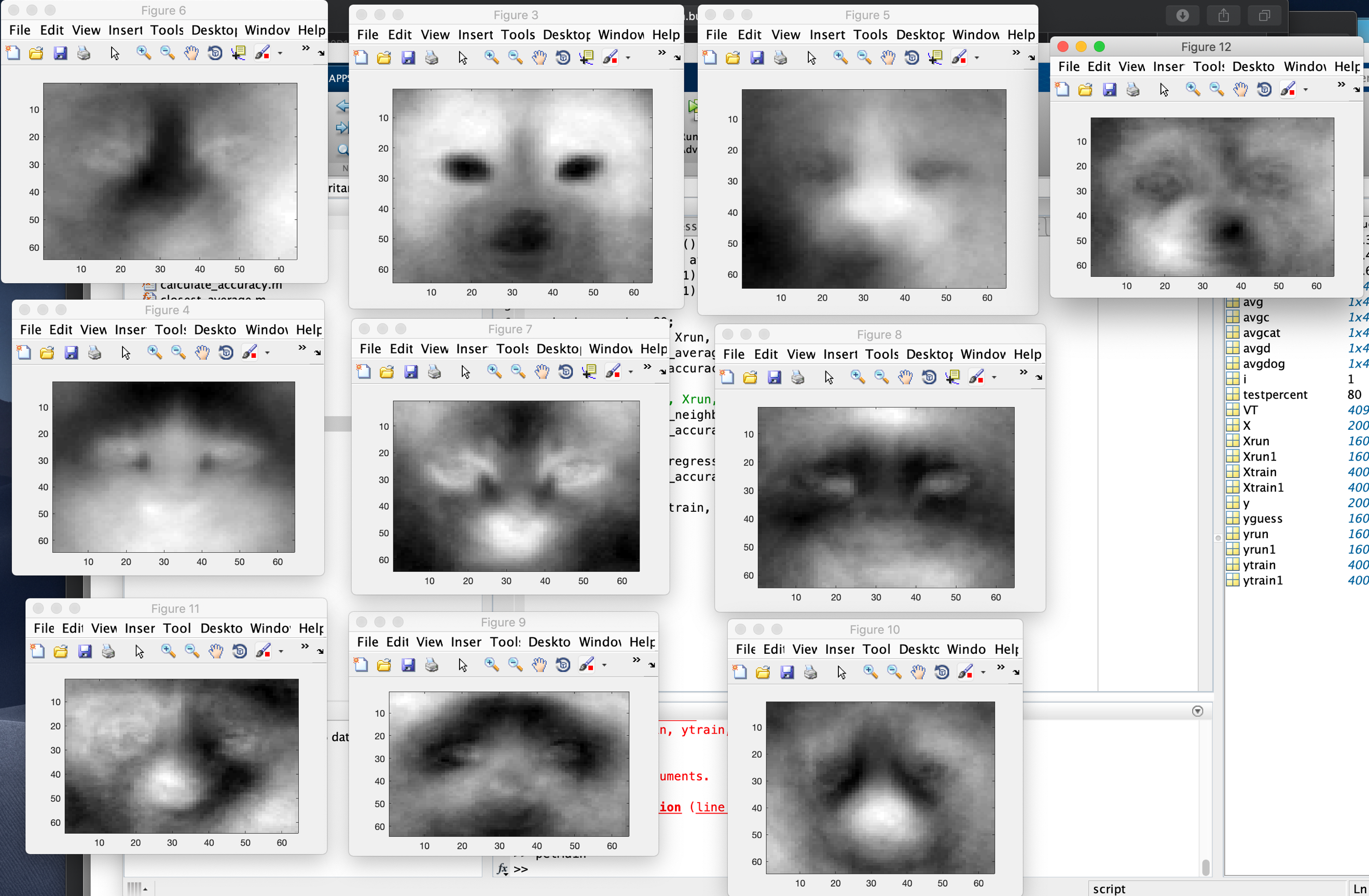
hlinX = sign(Xrun\*bols);

yguess = hlinX;

end



10.5



These pictures look like they kept the most important and telling features of classifying dog and cat.

10.6

%This function takes in a training data matrix Xtrain, training

%label vector ytrain and uses them to compute the PCA basis.

%It also takes in a data matrix Xrun and a dimension k

%and uses the top-k vectors in the PCA basis to reduce the

%dimension of Xtrain and Xrun. Finally, it uses the reduced data

%as inputs to the linear\_regression function to produce

%a vector of label guesses yguess.

function yguess = pca\_regression(Xtrain,ytrain,Xtest,k)

%FILL IN CODE

VT = pca(Xtrain);

VT = [VT zeros(4096,1)];

VT\_k = [VT zeros(4096,1)];

for i = 1:k

VT\_k(:,i) = VT(:,i);

end

Xtrainb = Xtrain\*VT\_k;

Xrunb = Xtest\*VT\_k;

bols = pinv((Xtrainb')\*Xtrainb)\*(Xtrainb')\*ytrain;

hlinX = sign(Xrunb\*bols);

yguess = hlinX;

end



main.m

[X, y] = read\_data();

[avgcat, avgdog] = average\_pet(X,y);

show\_image(avgcat,1);

show\_image(avgdog,1);

testpercent = 80;

[Xtrain, ytrain, Xrun, yrun] = split\_data(X,y,testpercent);

yguess = closest\_average(Xtrain,ytrain,Xrun);

acc = calculate\_accuracy(yrun',yguess);

%[Xtrain, ytrain, Xrun, yrun] = split\_data(X,y,testpercent);

yguess = nearest\_neighbor(Xtrain, ytrain, Xrun);

acc1 = calculate\_accuracy(yrun,yguess);

yguess = linear\_regression(Xtrain, ytrain, Xrun);

acc2 = calculate\_accuracy(yrun,yguess);

pca\_regression(Xtrain, ytrain, Xrun, 10);

accK10 = calculate\_accuracy(yrun,yguess);

pca\_regression(Xtrain, ytrain, Xrun, 20);

accK20 = calculate\_accuracy(yrun,yguess);

pca\_regression(Xtrain, ytrain, Xrun, 50);

accK50 = calculate\_accuracy(yrun,yguess);

pca\_regression(Xtrain, ytrain, Xrun, 100);

accK100 = calculate\_accuracy(yrun,yguess);