Here is a Matlab script that runs an example classifier. It is recommended that you run the script step by step (cut and paste), relating each step to the class slides. As you go along, answer the questions posed in red.

```matlab
%% Simulated data for a 1-D brain

clear all; close all; clc;

nVox=500; % number of voxels
nPerc=0.2; % percentage of 'active' voxels
nTrials=100; % total number of trials (per condition)
nTraining=90; % number of trials for training

% condition vector (class 1=0, class 2=1)
condition=zeros(2*nTrials,1);
condition(nTrials+1:end)=1;

% generate the design matrix
X(:,1)=1-condition; X(:,2)=condition;

% spatial activation pattern
spatial=rand(nVox,1);
spatial(spatial<1-nPerc)=0;
data=(spatial*condition').';
data=data+randn(size(data))*3;

figure(1);
imagesc(data); xlabel('voxels'); ylabel('trials');

%% estimate the beta vector and error variance
beta_hat=inv(X'*X)*X'*data;
Var_e=diag((data-X*beta_hat)'*(data-X*beta_hat)/(...
            nTrials-rank(X))).';

%% hypothesis testing; compute the t statistic
c=[-1;1];
t_stat=(c'*beta_hat)./sqrt(Var_e * (c'*inv(X'*X)*c));
t_thr=tinv(1-0.05/nVox,2*nTrials-rank(X));

figure(2);
subplot(2,1,1); plot(spatial);
ylabel('ground truth activation');
subplot(2,1,2); plot(t_stat);
xlabel('voxels');
ylabel('t-value');
hold on;
plot(1:length(t_stat),repmat(t_thr,[1 length(t_stat)]),'r:');
hold off;

%% split data (training / test)
idx0=find(condition==0);
idx1=find(condition==1);

Understand the way how the data is generated and also study the visualization in Figure 1. Do you (visually) recognize a difference between trials of the two conditions?

Is this high-dimensional data if you consider the voxels as features?

We first perform a GLM analysis (two-sample t-test) on the full data. Does any of the voxels survive statistical thresholding?

Then we split the data into training and test subsets.
traindata=1:nTraining;
testdata=nTraining+1:nTrials;

idxtest=[idx0(testdata); idx1(testdata)];
conditiontest=condition(idxtest);

%% Naive Bayes classifier
% learn model
mean0=mean(data(idx0(traindata),,:),1);
demeaned0=data(idx0(traindata),:)-repmat(mean0,[length(traindata) 1]);
covar0=(demeaned0.'*demeaned0)/length(traindata);
var0=diag(covar0).';

[-,eigs0]=eig(covar0);
figure(3);
subplot(2,1,1);
plot(diag(eigs0));
xlabel('index');
ylabel('eigenvalue');

mean1=mean(data(idx1(traindata),,:),1);
demeaned1=data(idx1(traindata),:)-repmat(mean1,[length(traindata) 1]);
covar1=(demeaned1.'*demeaned1)/length(traindata);
var1=diag(covar1).';

[-,eigs1]=eig(covar1);
subplot(2,1,2);
plot(diag(eigs1));
xlabel('index');
ylabel('eigenvalue');

% apply model
post0=sum((data(idxtest,:)-repmat(mean0,[2*(nTrials-nTraining) 1])).^2 ... 
./repmat(var0,[2*(nTrials-nTraining) 1]),2);
post1=sum((data(idxtest,:)-repmat(mean1,[2*(nTrials-nTraining) 1])).^2 ... 
./repmat(var1,[2*(nTrials-nTraining) 1]),2);

% error rate
err=sum(abs(conditiontest-(post1<post0)))/length(conditiontest);
fprintf('Error rate: %3.1f\n',err*100);

figure(4);
plot(conditiontest,'ko','MarkerSize',14);
hold on;
plot(post1<post0,'rx','MarkerSize',14);
xlabel('test trials');
ylabel('label');
legend({'ground truth','estimated'});
hold off;

We use a Naive Bayes (NB) classifier, which uses the marginal (Gaussian) distribution of the different features (voxels).

What are the parameters of the NB model?

Does the NB model use different type of information than the GLM above?

Study the eigenvalues of the empirical covariance matrix?
What do you observe? Try to estimate its inverse! Which additional assumption is made in NB to solve this problem?

Next, we evaluate the model for the test data by computing the posterior probability.

Run the code a couple of times. Is the NB classifier successful (on average) to retrieve the condition for (individual) test trials?

Do you expect more advanced classifiers that incorporate correlation structure between voxels to perform better for this data?