4.

$$MAE = \sum_{k=1}^{N} \frac{|\hat{l}_k - l_k|}{N} \quad (1)$$

Mean absolute error (MAE) is defined as the average of the absolute errors between the estimated ages and the ground truth age for the test image. The formula is shown above, where l_k is the ground truth age for the test image, $\hat{l}_k$ is the estimated age, and N is the total number of test images.

$$CS(j) = \frac{N_{e \leq j}}{N} \times 100\% \quad (2)$$

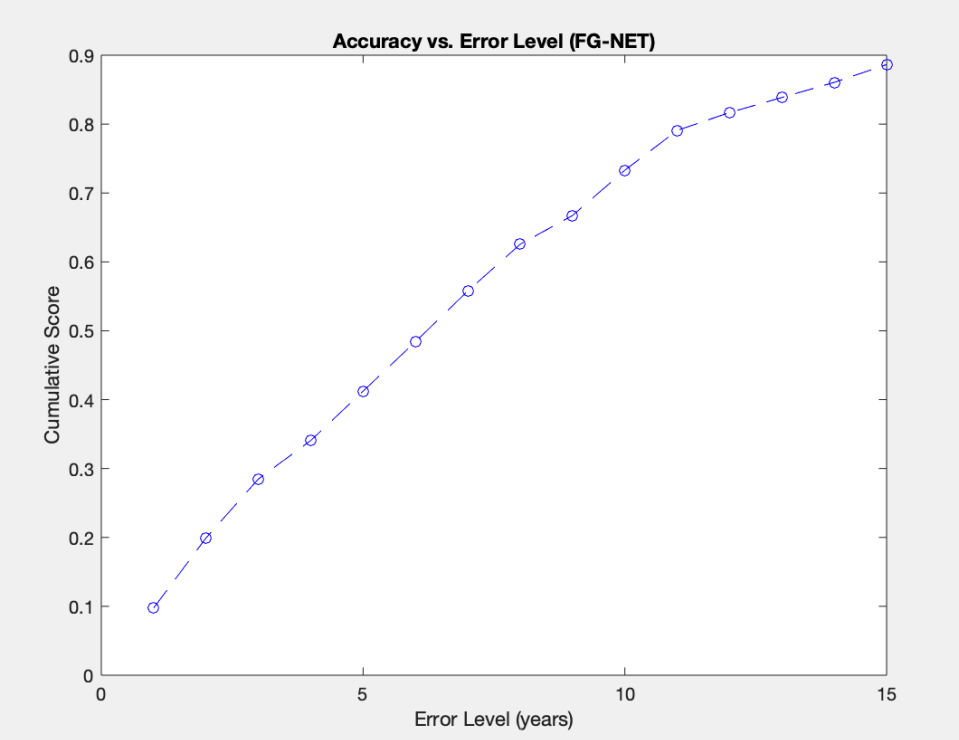
The cumulative score (CS) is defined as above, where $N_{e \leq j}$ is the number of test images on which age estimation makes an absolute error no higher than j years.

```
%% Compute the MAE and CS value (with cumulative error level of 5) for linear regres
MAE = mean(abs(yhat_test - ytest));

CS = sum (abs(yhat_test - ytest) <= err_level)/ nTest;
```

5.

```
%% generate a cumulative score (CS) vs. error level plot by varying the error level
CS_plot = zeros(15,1);
for err_level = 1:15
    CS_plot(err_level, 1) =  sum (abs(yhat_test - ytest) <= err_level)/ nTest;
end
figure
plot((1:15), CS_plot, 'b--o');
title('Accuracy vs. Error Level (FG-NET)');
xlabel('Error Level (years)');
ylabel('Cumulative Score');
```



From the plot above, it is clear that the cumulative score rises rapidly with the increase of the error level. In other words, when the large error level is used, the evaluation metric CS performs better.

6.
For partial least-squares regression, according to the official document of MATLAB, I choose the method below to train my model (parameters).

[XL,YL,XS,YS,BETA] = plsregress(X,Y,ncomp,...)

BETA is the model parameter vector, and it will be used to predict the test data (ages). Suppose X is an n-by-p matrix of predictor variables and Y is an n-by-m response matrix, BETA is a $(p+1)$-by-m matrix, containing intercept terms in the first row.

Y = [ones(n,1),X]*BETA + Yresiduals

Here $Yresiduals$ is the vector of response residuals, actually it is not necessary in my code. I use the method to predict the test data.

For the regression tree model, tree = fitrtree(X, Y) is used to train the model and use 'predict' method to predict the output (age).

```
%% Compute the MAE and CS value (with cumulative error level of 5) for both partial
err_level = 5;

% Partial least square regression
[~, ~, ~, ~, BETA] = plsregress(xtrain, ytrain);
yhat_test_pls = [ones(size(xtest,1),1) xtest]*BETA;

MAE_pls = mean(abs(yhat_test_pls - ytest));
CS_pls = sum (abs(yhat_test_pls - ytest) <= err_level)/ nTest;

% the regression tree model
tree = fitrtree(xtrain,ytrain);
yhat_test_tree = predict(tree,xtest);

MAE_tree = mean(abs(yhat_test_tree - ytest));
CS_tree = sum (abs(yhat_test_tree - ytest) <= err_level)/ nTest;
```

7.
First of all, we should select suitable model and kernel. As we are required to fit the data using support vector regression, so for '-s' parameter could be set as 3 or 4. According to the official guide of 'LIBSVM' toolbox [1], the radial basis function (RBF) kernel is recommended as first choice. There are 3 main advantages for choosing RBF kernel, it is nonlinear, relatively fewer hyperparameters and fewer numerical difficulties.

After determining model and kernel, we should find out the best parameters to achieve good performance. For an RBF kernel, the penalty parameter C and kernel parameter γ should be optimized, and in the guide cross-validation and grid-search are proposed. In my code, when using cross validation, I divide the training data into 10 folds of equal size ('-v 10').

How to determine the search range of the parameter when using grid-search? In theory, we could preset an enough large range, but it is impractical since plenty of time cost is required. Alternatively, we could change the search range multiple times by the result of the last run. For instance, I initially set the range of 'log2g' in the code from -1 to 1.5, but the result is that the best g is 0.5 (i.e. log2g is -1). That means the lower boundary of the 'log2g' is too large, thus I tune the lower boundary smaller.

Finally, in my case, the best parameter C is 256 and best γ is 0.03125, and the corresponding optimal MSE ('bestcv' in my code) is 55.8232. Note that when cross validation is used, the output of 'svmtrain' function is no longer a model, rather than the accuracy (for classification) or MSE (for regression).

```
%% Compute the MAE and CS value (with cumulative error level of 5) for Support Vecto
addpath(genpath('libsvm-3.14'));

% set the parameters via cross-validation! Elapsed time is 540.564505 seconds.
% bestc=256, bestg=0.03125, bestcv=55.8232
% bestc=0;bestg=0;
% bestcv=99999;
% tic
% for log2c = -1:10,
%   for log2g = -5:0.1:1.5,
%     cmd = ['-s 3 -v 10 -t 2 -c ', num2str(2^log2c), ' -g ', num2str(2^log2g)];
%     cv = svmtrain(ytrain, xtrain, cmd);
%     if (cv <= bestcv),
%       bestcv = cv; bestc = 2^log2c; bestg = 2^log2g;
%     end
%     fprintf('(best c=%g, g=%g, rate=%g)\n', bestc, bestg, bestcv);
%   end
% end
% toc
bestc = 256;
bestg = 0.03125;

options=sprintf('-s 3 -t 2 -c %f -g %f -b 1',bestc,bestg);
model=svmtrain(ytrain, xtrain, options);

[yhat_test_svr, ~, ~] = svmpredict(ytest,xtest, model);

err_level = 5;
MAE_svr = mean(abs(yhat_test_svr - ytest));
CS_svr = sum (abs(yhat_test_svr - ytest) <= err_level)/ nTest;
```

The results of CS and MAE for the four regression models.

CS	0.4124
cs_number	0
CS_plot	*15x1 double*
CS_pls	0.4084
CS_svr	0.6235
CS_tree	0.5040
database_path	'data_age.mat'
err_level	5
MAE	7.7044
MAE_pls	7.7103
MAE_svr	5.2369
MAE_tree	8.2350

	Linear regression	Partial least square regression	Regression tree model	Support vector regression
MAE	7.7044	7.7103	8.2350	5.2369
CS@5	0.4124	0.4084	0.5040	0.6235

From the table above, it is clear that SVR model performs better than the other three models in terms of both MAE and CS with an error level of 5.

Reference:

[1] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A practical guide to support vector classification." (2003): 1396-1400.

[2] Guo, Guodong, Yun Fu, Charles R. Dyer, and Thomas S. Huang. "Image-based human age estimation by manifold learning and locally adjusted robust regression." *IEEE Transactions on Image Processing* 17, no. 7 (2008): 1178-1188.