Cara Evangeline Chandra Mohan
190539421

MACHINE LEARNING FOR VISUAL DATA ANALYSIS LAB – 3

1. Compute the MAE and CS value (with a cumulative error level of 5) by comparing the estimated ages with the ground truth ages. You need to write your own code here.

```
%% Compute the MAE and CS value (with cumulative error level of 5) for linear regression
%------Mean Absolute Error-------
mae_LR = 0;
for i = 1:502     mae_LR = mae_LR + abs(ytest(i) - y_LR(i));
end
mae_LR = mae_LR/502;
%-----Cumulative Error----------
c = 0;
for i = 1:502
   error = abs(ytest(i) - y_LR(i));
   if error <= error_level
      c = c+1;
   end
end
cs_LR_5 = (c/502)*100;
```

Result:

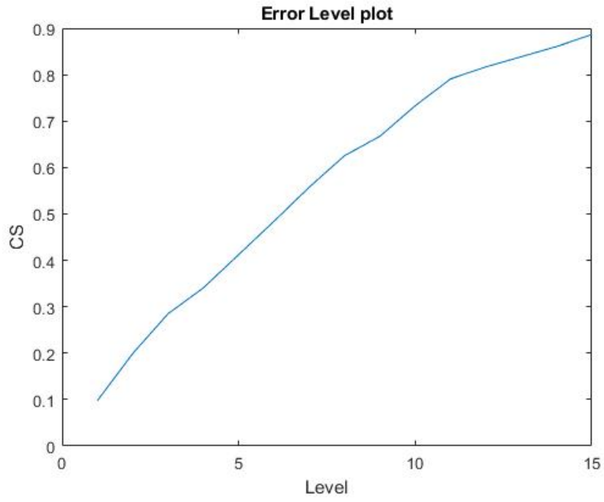
cs_LR_5 = 41.2351

mae_LR = 7.7044

Multiple Linear regression is used to train the model. Initially using regress(), weights of the model is found out and then applied on the test samples to get y_LR. Then MAE and CS is calculated as shown above.

2.)Vary the cumulative error level from 1 to 15 and generate a plot of the CS value against the cumulative error level. You need to write your own code here.

```
%% Generate a cumulative score (CS) vs. error level plot by varying the error level from
 1 to 15.
for error_level = 1:15
   c = 0;
   for i = 1:502
      error = abs(ytest(i) - y_LR(i));
      if error <= error_level
         c = c+1;
      end
   end
   cs_LR(error_level) = c/502;
end
plot(1:15,cs_LR);
```

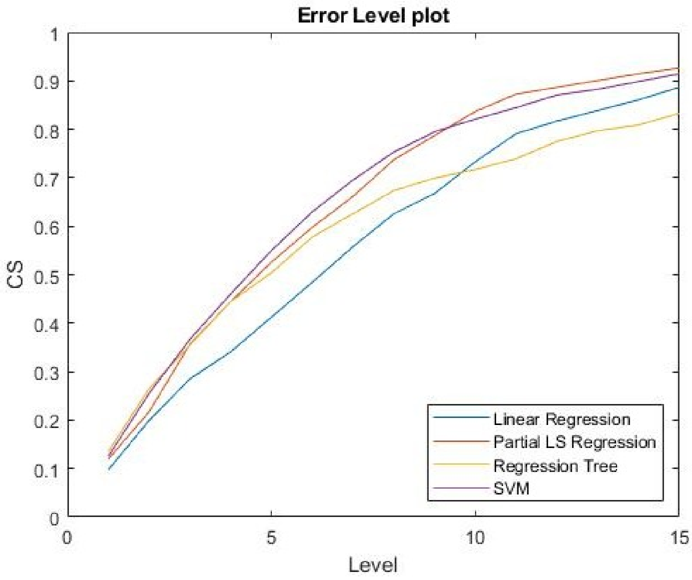


To generate the above graph, only linear regression model is considered and the error_level is iterated from 1 to 15 to get the corresponding Cumulative score.

Cara Evangeline Chandra Mohan
190539421

%%-------------Comparison between different models-------------------%%

```
for error_level = 1:15
   c1 = 0;
   c2 = 0;
   c3 = 0;
   c4 = 0;
   for i = 1:502
      error1 = abs(ytest(i) - y_LR(i));
      error2 = abs(ytest(i) - y_PLS(i));
      error3 = abs(ytest(i) - y_RT(i));
      error4 = abs(ytest(i) - y_SVM(i));
      if error1 <= error_level        c1 = c1+1;     end
      if error2 <= error_level        c2 = c2+1;     end
      if error3 <= error_level        c3 = c3+1;     end
      if error4 <= error_level        c4 = c4+1;     end
   end
   cs_LR(error_level) = (c1/502);
   cs_PLS(error_level) = (c2/502);
   cs_RT(error_level) = (c3/502);
   cs_SVM(error_level) = (c4/502);
end
plot(1:15,cs_LR);
hold on
plot(1:15,cs_PLS);
plot(1:15,cs_RT);
plot(1:15,cs_SVM);
hold off
legend('Linear Regression', 'Partial LS Regression', 'Regression Tree', 'SVM', 'Location',
'SouthEast')
xlabel('Level');
ylabel('CS');
title('Error Level plot');
```



The above graph (Graph 2) is a comparison of all the 4 models [Linear Regression, Partial Least Square Regression, Regression Tree and Support Vector Machine] and the above program is written after obtaining the predicted lables of all 4 models.

From the graph, we can infer that Regression Tree and SVM performs better while compared to Linear Regression and Partial Least Square Regression when taken for lower range of error_levels but overall SVM performs the best and then PLSR when we consider the entire graph up until higher error_level.

Cara Evangeline Chandra Mohan
190539421

Comparison between 4 models - MAE and CS values for error_level = 5

	Linear regression	PLSR	Regression Tree	SVM
MAE	*7.7044*	*6.0703*	*8.2350*	*6.0323*
CS_5	*41.2351*	*52.5896*	*50.3984*	*54.9801*

3.)Compute the MAE and CS values (with cumulative error level of 5) for both partial least square regression model and the regression tree model by using the Matlab built-in functions. You need to write your own code here.

```
%% Compute the MAE and CS value (with cumulative error level of 5) for both partial least square regression and the regression tree model by using the Matlab built in functions.
%%%%%----- Partial Least Square Regression -----%%%%%%%%%
[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xtrain,ytrain,10);
y_PLS = [ones(size(xtest,1),1) xtest]*beta;
%------Minimum Absolute Error-----------------
mae_PLS = 0;
for i = 1:502
   mae_PLS = mae_PLS + abs(ytest(i) - y_PLS(i));
end
mae_PLS = mae_PLS/502;
%-----Cumulative Error----------
c = 0;
for i = 1:502
   error = abs(ytest(i) - y_PLS(i));
   if error <= error_level
      c = c+1;
   end
end
cs_PLS_5 = (c/502)*100;
```

Result:

cs_PLS_5 = 52.5896
mae_PLS = 6.0703

10 PLS components(ncomp) are considered here, by default it takes min(size(xtrain,1)-1, size(xtrain,2)) which is 200 and this increases the error of the model, so after many considerations ncomp = 10 is taken.

```
%%%%%----- Regression Tree ---------%%%%%%%%%
Mdl = fitrtree(xtrain, ytrain);
y_RT = predict(Mdl, xtest);
%------Minimum Absolute Error-----------------
mae_RT = 0;
for i = 1:502
   mae_RT = mae_RT + abs(ytest(i) - y_RT(i));
end
mae_RT = mae_RT/502;
%-----Cumulative Error----------
c = 0;
for i = 1:502
   error = abs(ytest(i) - y_RT(i));
```

Result:

cs_RT_5 = 50.3984
mae_RT = 8.2350

The returned tree is a binary tree where each branching node is split based on the values of a column of xtrain.

```
    if error <= error_level
        c = c+1;
    end
end
cs_RT_5 = (c/502)*100;
```

4)Compute the MAE and CS values (with cumulative error level of 5) for Support Vector Regression by using the LIBSVM toolbox

```
%% Compute the MAE and CS value (with cumulative error level of 5) for Support Vector Regression by using LIBSVM toolbox
run('libsvm-3.14/matlab/make')
bestc=1024;bestg=2.8248; %Calculated best values after running the below part of the program [Marked in Green]
bestcv=0;
% tic
% for log2c = -1:10
%   for log2g = -1:0.1:1.5
%     cmd = ['-v 5 -t 1 -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('%g %g %g (best c=%g, g=%g, rate=%g)\n', log2c, log2g, cv, bestc, bestg, bestcv);
%   end
% end
% toc
options=sprintf('-s 4 -t 1 -c %f -b 1 -g %f -q', bestc, bestg);
model=svmtrain(ytrain, xtrain,options);
[y_SVM, accuracy , dec_values] = svmpredict(ytest,xtest, model,'-b 1');
%------Minimum Absolute Error-----------------
mae_SVM = 0;
for i = 1:502
    mae_SVM = mae_SVM + abs(ytest(i) - y_SVM(i));
end
mae_SVM = mae_SVM/502;
%-----Cumulative Error----------
c = 0;
for i = 1:502
    error = abs(ytest(i) - y_SVM(i));
    if error <= error_level
        c = c+1;
```

Result:

cs_SVM_5 = 54.9801

mae_SVM = 6.0323

```
    end
end
cs_SVM_5 = (c/502)*100;
```

The hyperparameters should be obtained initially before building the model, this is done by cross-validation. bestc and bestg values are obtained by iterating through many values and through the method of cross-validation of SVM and by the end of the iterations, bestc = 1024 and bestg = 2.8248 is obtained.
c is the cost of the model
g is the gamma in kernel function
s = 4 which refers to Support vector Regressor(SVR) and we use it here to predict continous value distributions
t = 1 Polynomial type kernel function
b = 1 Probability estimates [1 for SVR and 0 for SVC]

Hence we obtain the predicted labels of SVM model and calculate MAE and CS[error_level=5]