ASSIGNMENT 2
CSE 572 SPRING 2020

BY

SAI KASYAP KAMARAJU

Introduction:

This is assignment is an attempt understand human activities that can predict eating or non-eating activities based on the real-world Mayo data consists of Wristband Sensor, IMU and EMG sensors, but in this project only one sensor data is used for all users i.e EMG sensor which has a sampling rate of 100 Hz. The user dependent analysis is done in an in Phase 1; user independent analysis is done in phase 2. The Project is being on done MATLAB R2019b version.

Phase 1: User Dependent Analysis

1. **Procedure:**

   In assignment 1, the dimensionality reduction using PCA is done for each user based on the features obtained by Min, Max, FFT, Std and Mean. The new feature matrix that was obtained by multiplying the eigenvectors (PCA output) with the old feature matrix which is used for classification.

   Code Snippet: (In Assig2_Phase1_data.m)

```
% Creating New Feature Matrix
UsersEat_PCA = pca(UsersEat_Data);
UsersNotEat_PCA = pca( UsersNotEat_Data);
UsersEat_Newmat = UsersEat_Data * UsersEat_PCA(:,1:5);
UsersNotEat_Newmat = UsersNotEat_Data * UsersNotEat_PCA(:,1:5);
```

   The class imbalance problem in data is solved by taking equal number of samples of Eating and non – Eating for each user to properly train the data.

   After doing the above, the data was divided randomly (by eating and non-eating action each having equal number of samples) into 60% training data and 40% test data. Target data (Y) was produced by creating data matrices which have the same dimensions as the test and train data and classifying with a 1 or 0 based on the target as an eating or non-eating action.

   This is done as follows: In TrainTestSplit.m and Phase1_models.m

```
function[Data_Train,Data_Test]=TrainTestSplit(Data)
[m,n] = size(Data) ;
P = 0.60 ; %60 and 40 % split for train and Test
idx = randperm(m)  ;
Data_Train = Data(idx(1:round(P*m)),:) ;
Data_Test = Data(idx(round(P*m)+1:end),:) ;
end

%%%%Creating Train and Test data%% for ML Algorithms
[userEat_Train, userEat_Test] = TrainTestSplit(UsersEat_Newmat);
[userNotEat_Train, userNotEat_Test] =
TrainTestSplit(UsersNotEat_Newmat(1:size(UsersEat_Newmat, 1), :));
X_Train = [userEat_Train;userNotEat_Train];
X_Test = [userEat_Test; userNotEat_Test];
Y_Train = [ones(size(userEat_Train, 1), 1); zeros(size(userNotEat_Train, 1), 1)];
Y_Test = [ones(size(userEat_Test, 1), 1); zeros(size(userNotEat_Test, 1), 1)];
```

2. Machine Learning Algorithms

Decision Tree

Decision tree is built top-down from root node to break down dataset into smaller subsets. Each internal node of decision tree represents a "test" on an attribute and each leaf node represents the classification or decision. In the assignment, we have constructed Decision Tree using MATLAB function fitctree.

Snippet Code:

```
tree = fitctree(X_Train,Y_Train);
label_DT = predict(tree,X_Test);
```

Support Vector Machine

This is a supervised learning algorithm to construct the model to classify unlabeled data. In the assignment, we have constructed SVM model using MATLAB function fitcsvm to classify an action from rest of the other actions.

Snippet Code:

```
SVM = fitcsvm(X_Train,Y_Train,'Standardize',true,'KernelFunction','RBF',...
'KernelScale','auto');
label_SVM = predict(SVM,X_Test);
```

Neural Networks:

A Neural Network contains an input layer, a set of hidden layers and an output layer with varying number of neurons in each layer (input, output, hidden) depending on the application. In the assignment, we use the nntool of MATLAB and the output layer consists of only one neuron, which signifies the classification of each action. In this Project, Feed forward network of 10 layers is used.

Feedforward networks consist of a series of layers. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer produces the network's output.

Snippet Code:

```
% %%% Neural Network %%

net = feedforwardnet(10);
net = train(net, X_Train', Y_Train');
label_NN = sim(net, X_Test');
label_NN(label_NN>=0.5) = 1;
label_NN(label_NN<0.5) = 0;
```

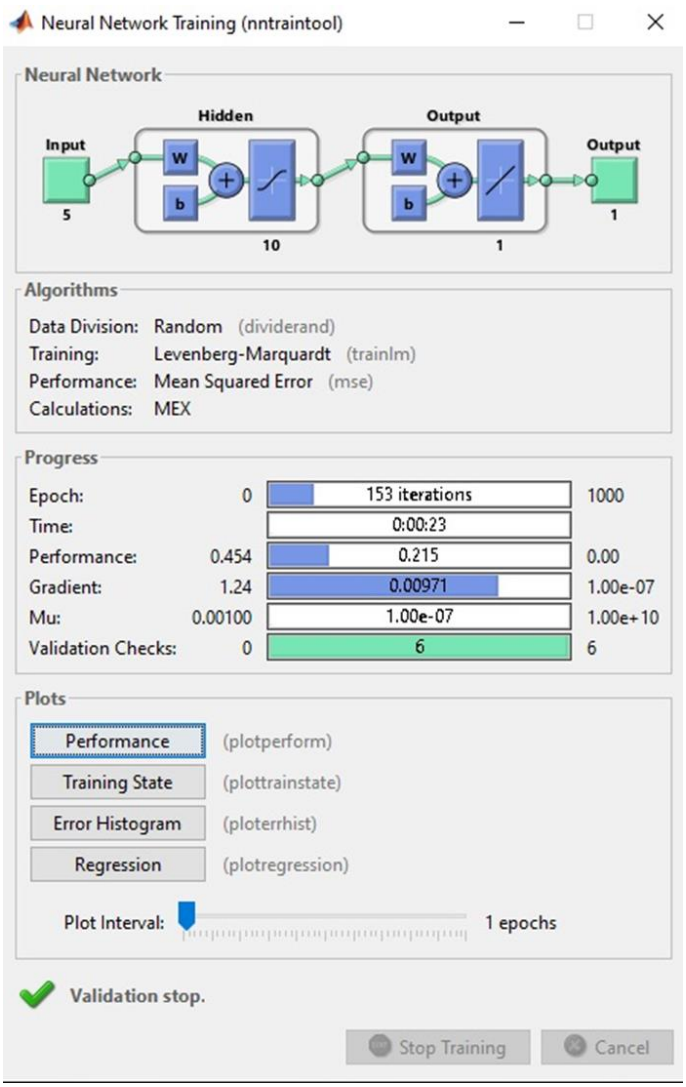

Figure 1 : Feed forward NN after training on all the 18 users

3. Performance Metrics:

the parameters which are used to measure the performance of machine learning algorithms and how to visualize them

Confusion Matrix

This matrix which is also called as error matrix is used to represent the values obtained on test data after predictions, from which the performance of a model can be evaluated. Below is the sample confusion matrix:

N= Number of Observations	Predicted -> Yes	Predicted -> No
Actual -> Yes	True Positive (TP)	False Positive (FP)
Actual -> No	False Negative (FN)	True Negative(TN)

Precision

It is the ratio of correctly predicted '+ve' classes to total number of classes which are classified as '+ve' classes. If the value of this performance metric is high, then it shows that the number of classes which are falsely classified as '+ve' classes is less. Precision = TP / (TP + FP)

Recall :

It is the ratio of correctly predicted '+ve' classes to all the actual '+ve' classes. It tells how well the model was able to classify the positive classes. Recall = TP / (TP + FN)

F1 Score:

It is weighted average of Precision and Recall. It is difficult to understand the concept from the definition, but it is helpful when class distribution is uneven. F1= (2 * Precision * Recall) / Precision + Recall.

Code Snippet for SVM Performance metrics:

```
[svm_confusion_matrix, order] = confusionmat(label_SVM, Y_Test);

% calculate precision, recall and fscore
for i=1:size(svm_confusion_matrix,1)
precision(i) = svm_confusion_matrix(i,i) / sum(svm_confusion_matrix(i,:));
end
precision(isnan(precision)) = [];
svm_precision = sum(precision) / size(svm_confusion_matrix, 1);

% calculate recall
for i=size(svm_confusion_matrix, 1)
recall(i) = svm_confusion_matrix(i,i) / sum(svm_confusion_matrix(:, i));
end

svm_recall = sum(recall)/size(svm_confusion_matrix, 1);
svm_fscore = 2 * svm_recall * svm_precision / (svm_precision + svm_recall);
```

4. **Results on Test Data :**

Results for Decision Tree:

User	Decision Tree FScore	Decision Tree Recall	Decision Tree Precision
User1	0.41638	0.82504	0.55344
User2	0.86279	0.81496	0.8382
User3	0.86903	0.79239	0.82894
User4	0.78564	0.74228	0.76334
User5	0.75034	0.68379	0.71552
User6	0.74018	0.67929	0.70843
User7	0.77382	0.67	0.71818
User8	0.73731	0.65102	0.69148
User9	0.70599	0.64365	0.67338
User10	0.8288	0.9016	0.86367
User11	0.87921	0.80473	0.84032
User12	0.79886	0.68588	0.73807
User13	0.76746	0.70141	0.73295
User14	0.67893	0.68953	0.68419
User15	0.57016	0.80891	0.66886
User16	0.78152	0.70093	0.73903
User17	0.86392	0.93592	0.89848
User18	0.82024	0.70787	0.75993
User19	0.81016	0.72722	0.76645
User20	0.75229	0.6635	0.70511
User21	0.80735	0.81227	0.8098
User22	0.84206	0.75273	0.7949
User23	0.90071	0.91199	0.90632
User24	0.84427	0.76608	0.80328
User25	0.87169	0.82507	0.84774
User26	0.90641	0.82691	0.86484
User27	0.78067	0.7095	0.74339
User28	0.73964	0.81414	0.7751
User29	0.89083	0.8197	0.85379
User30	0.79141	0.67074	0.7261

Test Results for SVM:

User	SVM FScore	SVM Recall	SVM Precision
User1	0.37348	0.86741	0.52215
User2	0.81988	0.84722	0.83332
User3	0.89528	0.81026	0.85065
User4	0.74323	0.80442	0.77262
User5	0.82371	0.74096	0.78015
User6	0.78418	0.71681	0.74898
User7	0.70975	0.71106	0.7104
User8	0.77901	0.6725	0.72185
User9	0.75834	0.68792	0.72142
User10	0.83748	0.89675	0.8661
User11	0.91077	0.83383	0.8706
User12	0.84142	0.69182	0.75932
User13	0.73183	0.70472	0.71802
User14	0.68367	0.69078	0.68721
User15	0.52901	0.8372	0.64834
User16	0.71866	0.74211	0.7302
User17	0.86287	0.91717	0.88919
User18	0.86053	0.74041	0.79597
User19	0.74994	0.75613	0.75302
User20	0.77061	0.6818	0.72349
User21	0.82127	0.83187	0.82653
User22	0.88129	0.78572	0.83077
User23	0.9066	0.90299	0.90479
User24	0.80699	0.81667	0.8118
User25	0.82481	0.86212	0.84306
User26	0.87112	0.8643	0.8677
User27	0.70117	0.73349	0.71697
User28	0.68669	0.86368	0.76508
User29	0.84436	0.85438	0.84934
User30	0.73578	0.67632	0.7048

Test Results for NN:

User	NN F1Score	NN Recall	NN Precision
User1	0.5458	0.39861	0.86531
User2	0.55932	0.40841	0.88709
User3	0.51955	0.38415	0.80236
User4	0.45558	0.3267	0.75241
User5	0.49417	0.36281	0.77461
User6	0.45444	0.3228	0.76741
User7	0.45299	0.32509	0.74676
User8	0.46147	0.33725	0.73058
User9	0.45772	0.33578	0.71871
User10	0.6046	0.44866	0.92669
User11	0.55888	0.41169	0.86989
User12	0.50647	0.37365	0.78577
User13	0.5088	0.39087	0.72864
User14	0.40604	0.33753	0.50944
User15	0.50706	0.3703	0.80402
User16	0.43478	0.3125	0.71429
User17	0.62384	0.46651	0.94131
User18	0.4631	0.32774	0.78895
User19	0.48888	0.35384	0.7906
User20	0.43588	0.31148	0.72572
User21	0.55273	0.40399	0.87482
User22	0.52446	0.38232	0.83486
User23	0.59821	0.44508	0.91195
User24	0.48732	0.3443	0.83361
User25	0.56409	0.41002	0.90366
User26	0.50277	0.36565	0.80446
User27	0.46951	0.35455	0.69479
User28	0.53979	0.38865	0.88329
User29	0.53996	0.39312	0.86186
User30	0.44693	0.31872	0.7477

Interpretation: The best ml model depends on the user and performance metric to evaluated for the user as the data (EMG) varies for each user.

Phase 2: User Independent Analysis (Code in Phase2_Indep.m)

1. **Procedure:**

In this analysis, 60 % of total users i.e users 1 to 18 are considered for training the ML models. While the remaining models i.e 19 to 30 are considered as test.

In both training and test data, equal number of eating and non-eating samples are taken to eliminate the class imbalance and to improve the performance of algorithms.

Code Snippet: Code in Phase2_Indep.m

```
for t = 1 : 18
%%% % eating
data%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
userData = userList(contains(userList,userNums(t,:)));
UsersEat_Newmat = userData(contains(userData,'UsersEat_Newmat'));
UsersNotEat_Newmat = userData(contains(userData,'UsersNotEat_Newmat'));
% Load  .mat files
load(strcat(output,'\',UsersEat_Newmat{1}));
% Load  .mat files

% Load .mat files
load(strcat(output,'\',UsersNotEat_Newmat{1}));

userEat_Train = [userEat_Train;UsersEat_Newmat];
userNotEat_Train =[userNotEat_Train;UsersNotEat_Newmat(1:size(UsersEat_Newmat, 1), :)];

end
X_Train = [userEat_Train;userNotEat_Train];

Y_Train = [ones(size(userEat_Train, 1), 1); zeros(size(userNotEat_Train, 1), 1)];
```

The testing data is a loop of each user from user 19 to user 30.

2. **ML algorithms selected are the same that were used in Phase 1.**
3. **The Code for all 3 algorithms is also the same but they are trained prior in this case for all the 18 users.**

Code snippet:

```
% %%%%% SVM %%%%%%%%%%%%%%%%%%%%%%%%%%%%
SVM=fitcsvm(X_Train,Y_Train,'Standardize',true,'KernelFunction','RBF',...
'KernelScale','auto');
%%%Decision Tree%%%%%%
tree = fitctree(X_Train,Y_Train);

% %%% Neural Network %%
```

net = feedforwardnet(10);
net = train(net, X_Train', Y_Train');

But are tested for each user ranging from 19 to 30.

4. **Results:**

Test Results for Decision Tree

User	Decision Tree FScore	Decision Tree Recall	Decision Tree Precision
User19	0.24896	0.57766	0.34796
User20	0.65419	0.5592	0.60298
User21	0.69516	0.64746	0.67046
User22	0.78377	0.6285	0.6976
User23	0.6583	0.58833	0.62135
User24	0.62497	0.58908	0.6065
User25	0.74377	0.61733	0.67468
User26	0.7878	0.66756	0.72271
User27	0.69506	0.58369	0.63453
User28	0.672	0.61391	0.64164
User29	0.68765	0.59973	0.64069
User30	0.73165	0.56952	0.64048

Test Results for SVM:

User	SVM FScore	SVM Recall	SVM Precision
User19	0.25586	0.53064	0.34525
User20	0.71844	0.53789	0.61519
User21	0.75574	0.69429	0.72371
User22	0.84768	0.69657	0.76474
User23	0.69476	0.61525	0.65259
User24	0.64264	0.58869	0.61448
User25	0.79388	0.63988	0.70861
User26	0.84359	0.69329	0.76109
User27	0.69157	0.55091	0.61328
User28	0.71217	0.56281	0.62874
User29	0.73856	0.63317	0.68181
User30	0.78924	0.62282	0.69622

Test Results for NN:

User	NN F1Score	NN Recall	NN Precision
User19	0.34983	0.23503	0.68391
User20	0.30371	0.20163	0.61514
User21	0.49557	0.35334	0.82942
User22	0.48787	0.36008	0.75626
User23	0.41621	0.29641	0.69858
User24	0.34476	0.22306	0.7587
User25	0.452	0.31272	0.815
User26	0.46869	0.33248	0.79396
User27	0.32529	0.21903	0.63181
User28	0.42744	0.29803	0.75548
User29	0.41431	0.28038	0.79321
User30	0.42715	0.30851	0.69405

Conclusion

In the User Dependent Analysis, the model is trained and tested for each user, where as in user independent analysis the model is trained for 60 % users there by making model to generalize better. Also, the User dependent and Independent analysis of this data using Support Vector Machines, Decision Trees and Neural Network Machines the Precision, Recall and F1-scores varied depending on the amount of test and training data. Our observation is that the model data was affected by the feature selection and PCA results and since the IMU data was not considered. The variance that the IMU data would have provided could have shown better results for the above binary classification models.

References :

1. https://www.mathworks.com/matlabcentral/answers/395136-how-to-divide-a-data-set-randomly-into-training-and-testing-data-set
2. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
3. https://www.mathworks.com