Project Assignment 2 and 3 Report

DATA MINING

CSE 572: Spring 2018

Submitted to:

Professor Ayan Banerjee
Ira A. Fulton School of Engineering
Arizona State University

Submitted by(GROUP 20):

Vivek Agarwal (vagarw14@asu.edu)
Suchita Vichare (ssvichar@asu.edu)
Supriyaa Damodaraswamy (sdamoda3@asu.edu)
Sukanya Panda (spanda7@asu.edu)
Ayush Ray (aray29@asu.edu)

November 30, 2018

1. General Procedure

We are using IMU data for Eating Activity using Fork for Assignment 2 and 3. During the previous task we had performed data aggregation using features like Mean, Root Mean Square, Maximum, Variance, Variance of Fast Fourier Transform. We have selected more discriminating features by performing Principal Component Analysis on the normalized feature matrix. Continuing forward from the previous task we are utilizing IMU data in transformed vector space using PCA for 30 users. We are now performing classification to classify the data into 2 labels : Eating and Non Eating. Eating activity data is fetched using Ground Truth File and remaining data is labelled as Non Eating.

For classification, we perform these following tasks for Assignment 2 and Assignment 3 respectively:

1. We have divided data into Training set and Testing Set with 60% and 40% ratio for each user. We then train the data on 3 separate classification models : Support Vector Machine, Decision Trees and Neural Networks and then evaluate performance of the classification models and report Accuracy using metrics like Precision, Recall and F-score.
2. We have divided data into 60 % users for training and rest 40 % users will be tested on the classification machines : Support Vector Machine, Decision Trees and Neural Networks. Each Test user will be evaluated individually. All the performance metrics used are same as assignment 2.

2. Performance Metrics

Different machine Learning Algorithms are being implemented on the data sets. Hence, we take into consideration different parameters that are used to measure the performance of the Machine Learning Algorithms.

2.1.Confusion Matrix

Confusion matrix is also called the Error Matrix. The Matrix is a visualization of the performance of the algorithm. The confusion Matrix is used to summarize the results of testing the algorithm for further interrogation.

An example of a Confusion Matrix:

	Class = Yes	Class = No
Class = Yes	True Positive (TP)	False Negative (FN)
Class = No	False Positive (FP)	True Negative (TN)

True Positive (TP): The total number of positive classes which have been correctly identified, that is the class which is positive (+ve) has been classified as positive (+ve) by the model.

True Negative (TN): The total number of negative classes which have been correctly identified, that is the class which is negative (-ve) has been classified as negative (-ve) by the model.

False Positive (FP): The class is negative (-ve) but is classified as positive (+ve) by the model.

False Negative (FN): The class is positive (+ve) but is classified as negative (-ve) by the model.

2.2.Precision, Recall, F1 and Accuracy:

After understanding the above four parameters, we can calculate Accuracy, Precision, Recall and F1 score through the True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

2.2.1.Precision

Precision can be defined as the ratio of correctly predicted positive classes to the total number of predicted classes

A higher value of the performance metric represents that number of classes which are falsely classified as positive is less.
Precision = TP/(TP + FP)

2.2.2.Recall

Recall can be defined as the ratio of total number of positive classes to the total number of predicted classes. The value of Recall indicates how well the model was able to classify positive classes.
Recall = TP/(TP+FN)

2.2.3. F_Score

F_Score can be defined as the weighted average of Precision and Recall.

F_Score takes into account both False Positives and False Negatives. It is more significant than accuracy in case the distribution of classes in not even, whereas Accuracy works well if the False Positive classes and False Negative classes have the similar cost. We have to take into account both Precision and Recall if the False Positives and False Negatives classes have almost same.
F_Score = 2*(Recall * Precision)/(Recall + Precision)

2.2.4.Accuracy

Accuracy can be defined as the measure of correctly predicted classes to the total number of classes.
Accuracy is best measured when on datasets which are symmetric where we have almost the same values for False Negative and False Positive Classes.
Accuracy = (TP+TN)/(TP+FP+FN+TN)

While building a model, we need to take into account the above parameters to find out the performance of the model.

2.2.5 Receiver Operating Characteristic (ROC) curve

In a Receiver Operating Characteristic (ROC) curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for

different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve (AUC) is a measure of how well a parameter can distinguish between two diagnostic groups.

2.2.6 Cross Entropy

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as: $-(y\log(p)+(1-y)\log(1-p))$

3. Implementation of Machine Learning Algorithms

Implemented the following machine learning algorithms to classify eating from non-eating activities:

- Decision Tree
- Support Vector Machine
- Neural Network

3.1. Training and Test Data

In Assignment 2 we perform User specific Classification while in Assignment 3 for User Independent.

For User Specific:

Dimension of Combined data : 59200 x 10
Dimension of Combined aggregated data after PCA: 2960 x 10
Dimension of data (per user) after PCA: ~ 100 x 10
Dimension of Training data (per user) after PCA: ~ 60 x 10
Dimension of Test Data (per user) after PCA: ~ 40 x 10

For User Independent:

Dimension of Training data: 1780 x 10
Dimension of Test Data: 1140 x 10

3.2. Decision Tree

Decision tree is a top-down tree. It is built from the root node to the leaf node. This is done to break down datasets into smaller subtasks. Each internal node of the tree represents a "test" on an attribute and each leaf node of the tree represents the classification or decision. In this assignment, we have constructed the decision tree using the function **fitctree** in matlab.

3.2.1. USER SPECIFIC TEST RESULTS FOR DECISION TREE (Assignment 2)

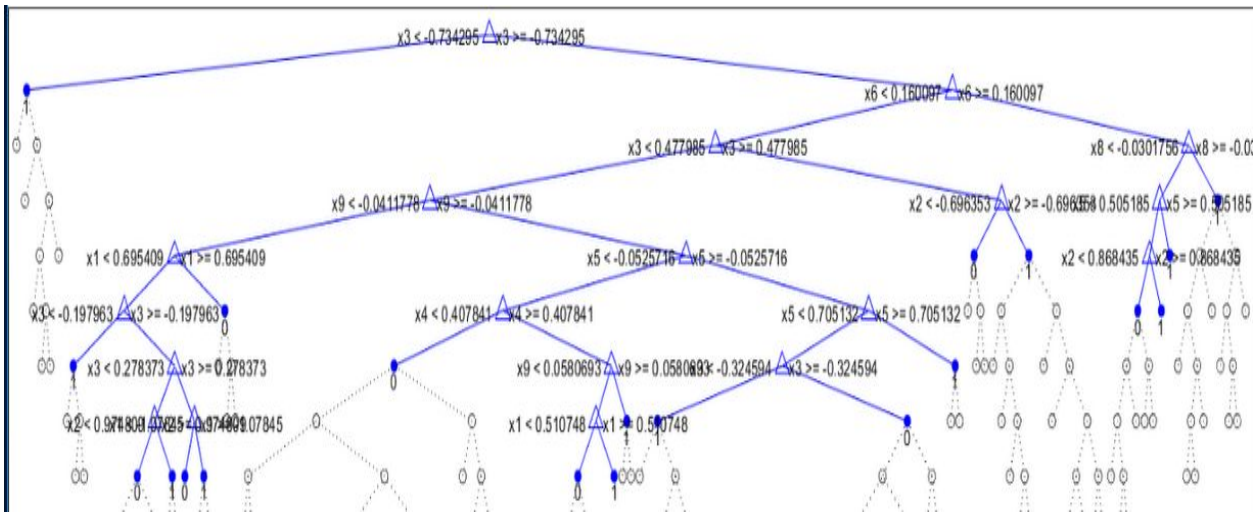


Fig. Decision Tree graph (after pruning)

User	Recall	Precision	F_score	Accuracy
1	0.8	0.857143	0.827586	80
2	0.825	0.838542	0.831716	82.5
3	0.641667	0.644928	0.643293	64.1667
4	0.63125	0.631765	0.631507	63.125
5	0.6	0.606112	0.603041	60
6	0.579167	0.581202	0.580183	57.9167
7	0.589286	0.592735	0.591005	58.9286

8	0.575	0.578125	0.576558	57.5
9	0.588889	0.604727	0.596703	58.8889
10	0.6525	0.655775	0.654133	65.25
11	0.643182	0.64735	0.645259	64.3182
12	0.60625	0.610121	0.60818	60.625
13	0.607692	0.610303	0.608995	60.7692
14	0.6125	0.617727	0.615102	61.25
15	0.584459	0.599169	0.591723	58.4459
16	0.617089	0.618015	0.617552	61.7089
17	0.61756	0.619336	0.618447	61.756
18	0.599719	0.601809	0.600762	59.9719
19	0.590426	0.597308	0.593847	59.0426
20	0.588384	0.595205	0.591775	58.8384
21	0.557692	0.565797	0.561715	55.7692
22	0.557339	0.56762	0.562433	55.7339
23	0.557018	0.566998	0.561964	55.7018
24	0.536765	0.54282	0.539775	53.6765
25	0.568548	0.580929	0.574672	56.8548
26	0.570039	0.581674	0.575798	57.0039
27	0.557116	0.563308	0.560195	55.7116
28	0.546533	0.550036	0.548279	54.6533
29	0.564716	0.570458	0.567573	56.4716
30	0.552226	0.557776	0.554987	55.2226

3.2.2. USER INDEPENDENT TEST RESULTS FOR DECISION TREE (Assignment 3)

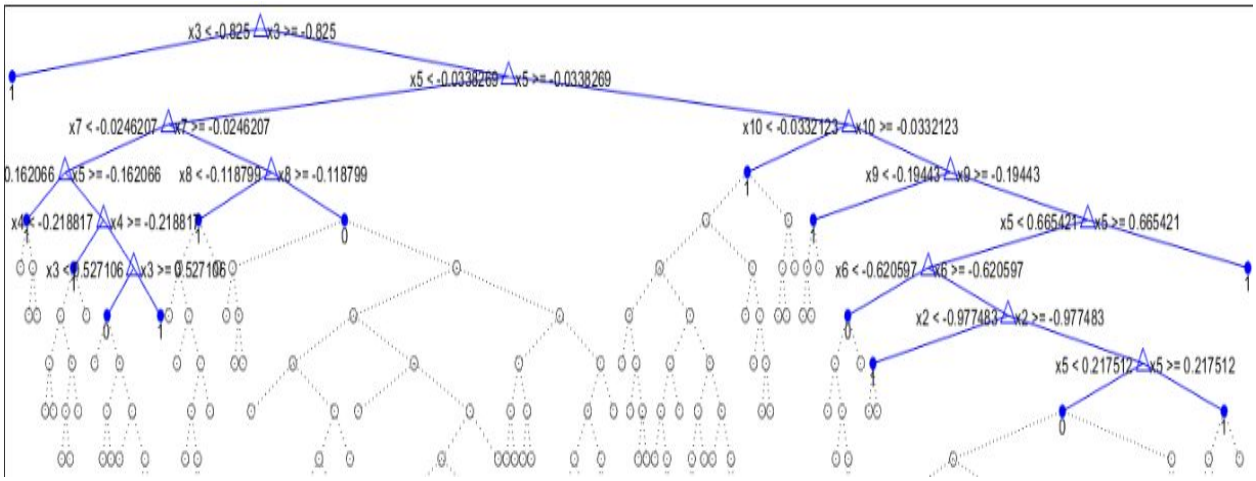


Fig. Decision Tree graph (after pruning)

Test No.	Recall	Precision	F_score	Accuracy
1	0.5	0.5	0.5	54
2	0.535	0.537333	0.536164	57
3	0.516667	0.517626	0.517146	52.6667
4	0.51	0.510101	0.510051	47.5
5	0.526	0.52602	0.52601	46.4
6	0.538333	0.538385	0.538359	48.8333
7	0.558571	0.558629	0.5586	51.7143
8	0.575949	0.57595	0.57595	51.3924
9	0.598876	0.598937	0.598907	52.9213
10	0.6125	0.612831	0.612665	54.0625
11	0.591346	0.591495	0.591421	52.9808
12	0.609649	0.609747	0.609698	51.5789

3.3. Support Vector Machine

This is a supervised machine learning algorithm which is used to construct a model that will classify the given activity data into eating and non-eating. We used MATLAB ***fitcsvm*** to build the SVM model which classifies the actions into eating and non-eating.

3.3.1. USER SPECIFIC TEST RESULTS FOR SVM: (Assignment 2)

User No.	Recall	Precision	F_Score	Accuracy
1	0.925	0.934783	0.929866	92.5
2	0.825	0.87037	0.847078	82.5
3	0.766667	0.76936	0.768011	76.66667
4	0.6	0.63805	0.61844	60
5	0.625	0.62872	0.626854	62.5
6	0.55	0.560504	0.555202	55
7	0.571429	0.585034	0.578151	57.14286
8	0.575	0.586346	0.580618	57.5
9	0.538889	0.544929	0.541892	53.88889
10	0.545	0.555208	0.550057	54.5
11	0.565909	0.582645	0.574155	56.59091
12	0.560417	0.584963	0.572427	56.04167
13	0.551923	0.563496	0.55765	55.19231
14	0.55	0.559935	0.554923	55

15	0.550676	0.565279	0.557882	55.06757
16	0.556962	0.570386	0.563594	55.6962
17	0.558036	0.572242	0.56505	55.80357
18	0.570225	0.589736	0.579816	57.02247
19	0.573138	0.593363	0.583075	57.31383
20	0.579545	0.60642	0.592678	57.95455
21	0.569712	0.599643	0.584294	56.97115
22	0.567661	0.60041	0.583576	56.76606
23	0.563596	0.597155	0.579891	56.35965
24	0.556723	0.58363	0.569859	55.67227
25	0.568548	0.602692	0.585122	56.85484
26	0.561284	0.600468	0.580215	56.1284
27	0.564607	0.606534	0.58482	56.46067
28	0.565693	0.612739	0.588277	56.56934
29	0.570922	0.617893	0.59348	57.0922
30	0.552226	0.595452	0.573025	55.2226

3.3.2. USER INDEPENDENT TEST RESULTS FOR SVM: (Assignment 3)

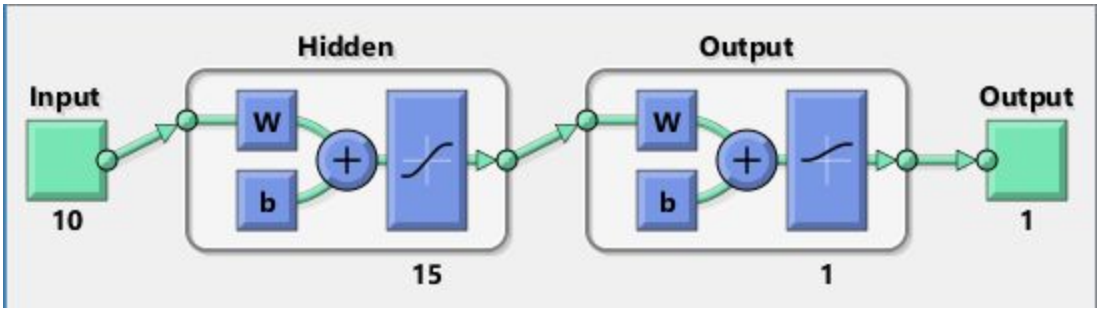
Test No.	Recall	Precision	F_Score	Accuracy
1	0.59	0.594578	0.59228	54
2	0.74	0.74	0.74	57
3	0.753333	0.758903	0.756108	52.66667
4	0.7	0.722816	0.711225	47.5
5	0.706	0.737463	0.721389	46.4
6	0.713333	0.74	0.726422	48.83333
7	0.71	0.743012	0.726131	51.71429
8	0.708861	0.736023	0.722187	51.39241
9	0.720225	0.749748	0.73469	52.92135
10	0.723958	0.755649	0.739464	54.0625
11	0.565909	0.582645	0.574155	52.98077
12	0.707895	0.72641	0.717033	51.57895

Code Snippet for SVM and Decision Trees

```
svm_model =  fitcsvm(train_data,train_data_label);
tree_model = fitctree(train_data,train_data_label);
svm_fit_data = predict(svm_model,test_data);
tree_fit_data = predict(tree_model,test_data);
predictedList = {tree_fit_data,svm_fit_data};

for k = 1:2
    confMat = confusionmat(test_data_label,predictedList{k});
    Accuracy = [];
    recall = [];
    precision = [];
    for j =1:size(confMat,1)
        recall(j)=confMat(j,j)/sum(confMat(j,:));
    end

    Recall=sum(recall)/size(confMat,1);
    for j =1:size(confMat,1)
        precision(j)=confMat(j,j)/sum(confMat(:,j));
    end

    Precision=sum(precision)/size(confMat,1);

    F_score=2*Recall*Precision/(Precision+Recall);
    TP = confMat(1,1);
    TN = confMat(2,2);
    FP = confMat(1,2);
    FN = confMat(2,1);

    Accuracy = [Accuracy; 100*(TP+TN)/(TP+TN+FP+FN)];
    tempRow = [Recall,Precision,F_score,Accuracy];
    if k == 1
        DtreeRecallPrecisionMatrix  = [DtreeRecallPrecisionMatrix ;tempRow];
    else
        SvmRecallPrecisionMatrix  = [SvmRecallPrecisionMatrix ;tempRow];
    end
end
```

3.4. Neural Network

A Neural Network is made up of an input layer, an output layer and a set of hidden layers. The number of neurons is varying in every layer based on the application. For this project, we use the neural network deep-learning toolbox of MATLAB and one neuron is generated in the output layer.



3.4.1. Input to Patternet

Patternet (Pattern recognition networks) classify the inputs given to them according to the target classes specified, which are the feed-forward networks. The input to the patternnet is the number of neurons in the hidden layer.

3.4.2. Model Parameters

Two parameters are passed:
i) Training Function: ‘trainscg’ : network training function that updates weight and bias values according to the scaled conjugate gradient method.
ii) Performance Function: ‘mse' : Mean Square Error

3.4.3. Model Details with code snippets

Number of neurons in input layer: 10
Number of neurons in hidden layer: 15
Number of neurons in the output layer: 1
Number of layers (input, hidden, output): 1

Code Snippet for Neural Net:

```
hiddenLayer = 15;

trainFcn = 'trainscg';
net.performFcn = 'mse';
net = patternnet(hiddenLayer, trainFcn);


% Train the Network
[net,tr] = train(net,user_data,user_data_label);

% Test the Network
y_output = net(user_data);

[cval,cmval,indicat,per] = confusion(user_data_label,y_output);
TP = cmval(1,1);
TN = cmval(2,2);
FP = cmval(1,2);
FN = cmval(2,1);

Precision = [Precision ; TP / (TP + FP )];
Recall = [Recall ; TP / ( TP + FN )];
F1= [F1;2*TP/(2*TP+FP+FN)];
Accuracy = [Accuracy; (TP+TN)/(TP+TN+FP+FN)];
```

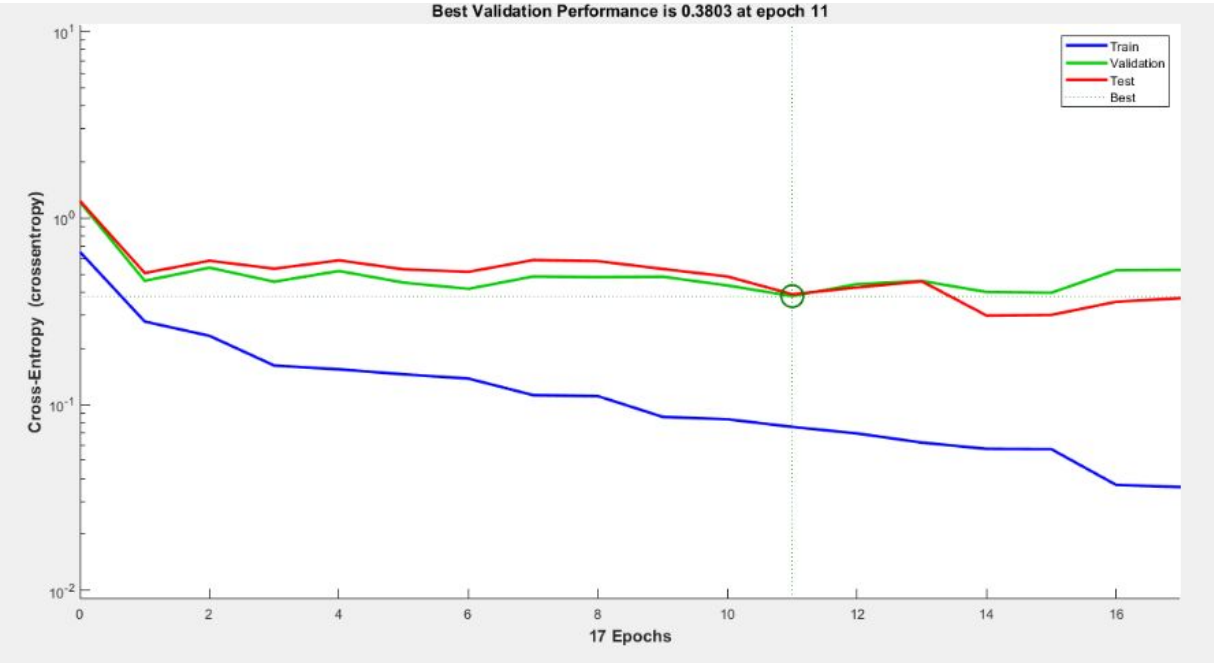
3.4.4. USER SPECIFIC TEST RESULTS FOR NEURAL NETWORK:

(Assignment 2)

User No.	Recall	Precision	F_Score	Accuracy
1	0.98	0.98	0.98	98
2	0.78	0.78	0.78	78
3	0.957447	0.9	0.927835	93
4	0.505051	1	0.671141	51
5	0.816327	0.8	0.808081	81
6	0.816667	0.98	0.890909	88
7	0.98	0.98	0.98	98
8	0.980392	1	0.990099	99
9	0.674419	0.58	0.623656	65

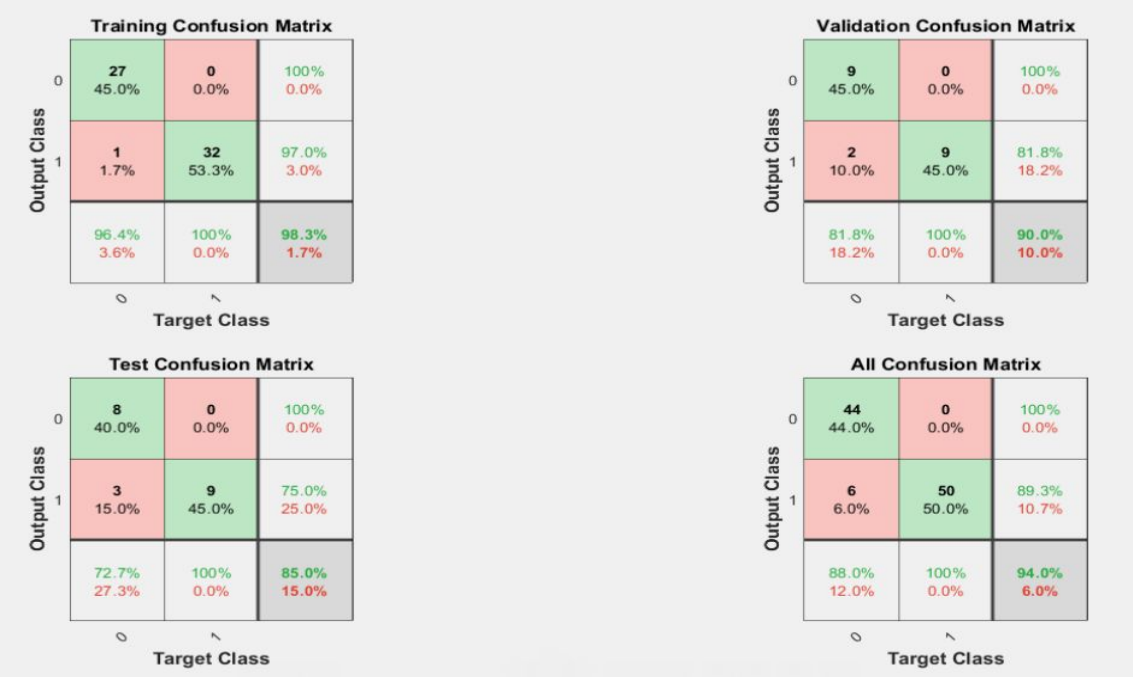
10	0.824561	0.94	0.878505	87
11	0.94	0.94	0.94	94
12	0.757576	1	0.862069	84
13	0.921569	0.94	0.930693	93
14	0.803571	0.9	0.849057	84
15	0.714286	0.875	0.786517	76.25
16	0.740741	0.8	0.769231	76
17	0.86	0.86	0.86	86
18	0.758065	0.94	0.839286	82
19	0.741935	0.92	0.821429	80
20	0.851852	0.92	0.884615	88
21	0.754098	0.92	0.828829	81
22	0.92	0.92	0.92	92
23	0.745455	0.82	0.780952	77
24	0.803922	0.82	0.811881	81
25	0.882353	0.9	0.891089	89
26	0.8	0.977778	0.88	86.6667
27	0.818182	0.9	0.857143	85
28	0.555556	0.142857	0.227273	51.4286
29	0.95	0.95	0.95	95
30	0.923077	0.96	0.941176	94

3.4.4.1. PERFORMANCE EVALUATION:

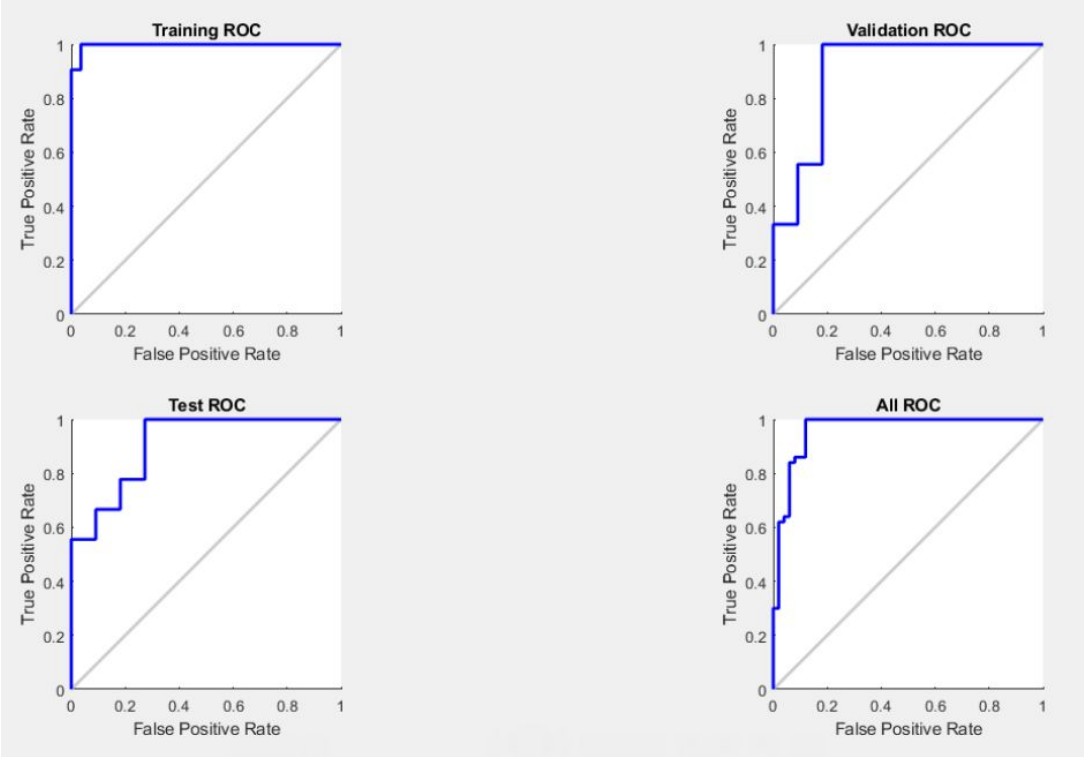


MSE measures the average of squared difference between the predicted values and actual target values

3.4.4.2. CONFUSION MATRIX:



3.4.4.3. Receiver Operating Characteristics:

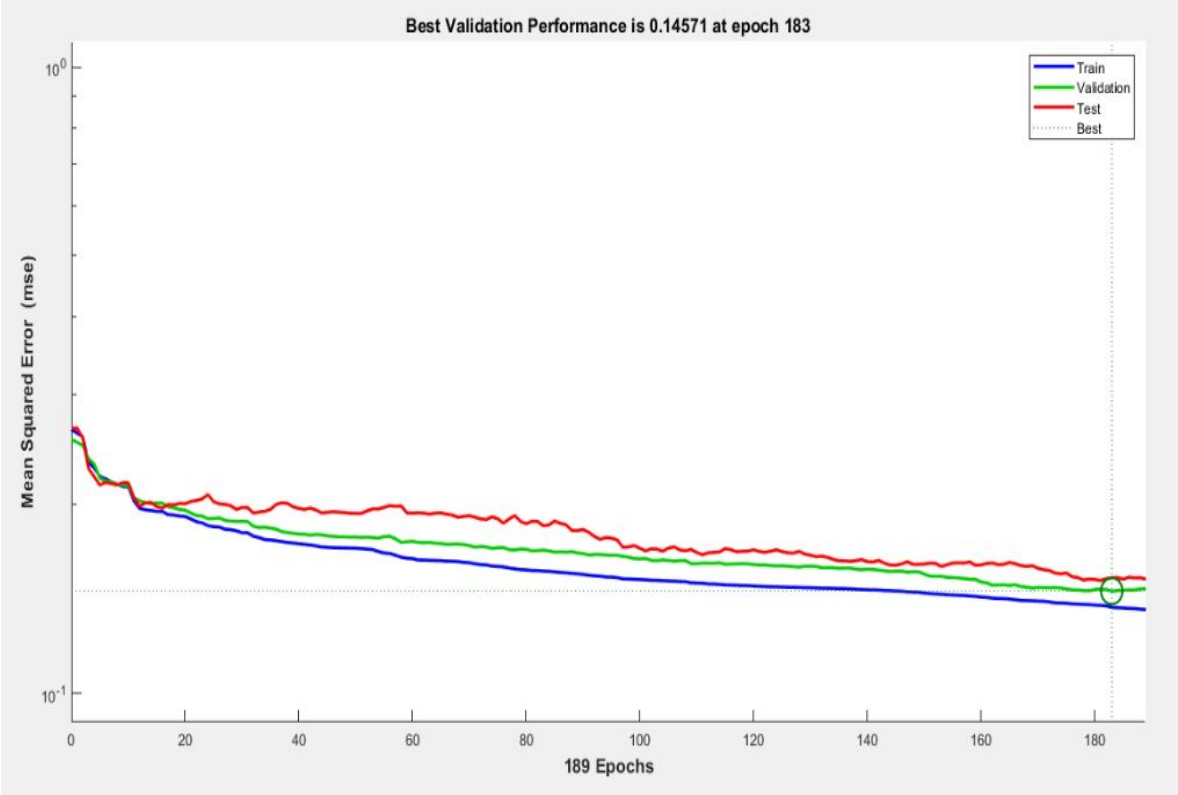


3.4.5. USER INDEPENDENT TEST RESULTS FOR NEURAL NETWORK:

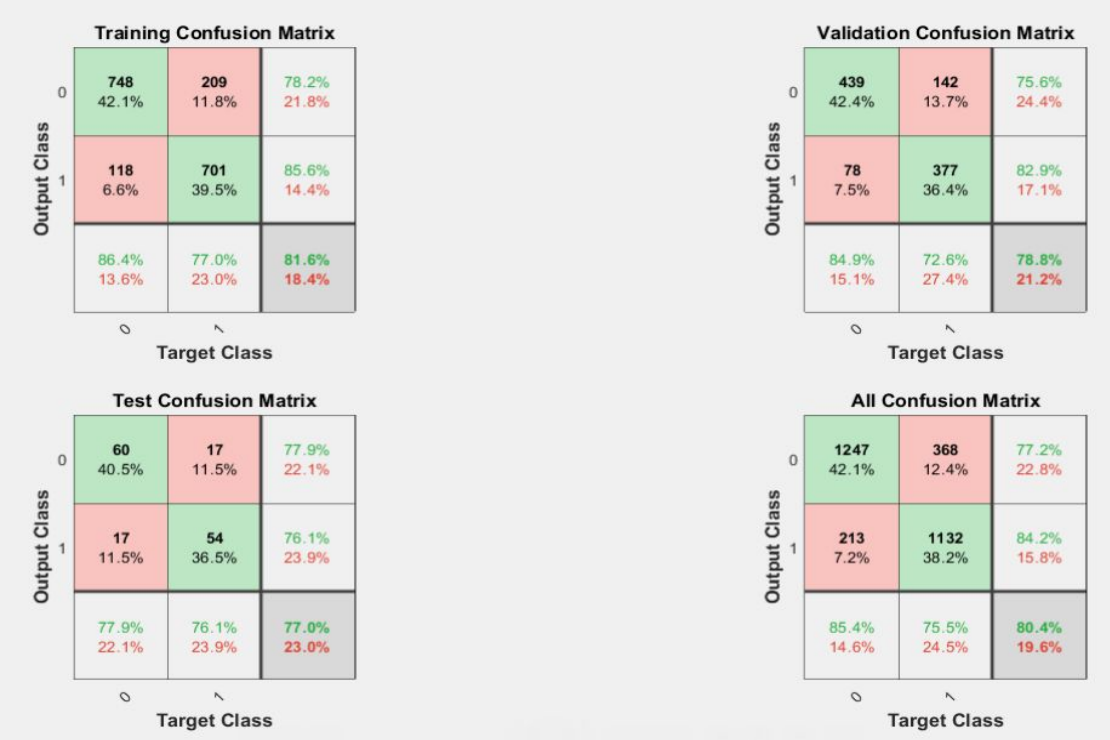
(Assignment 3)

User No.	Recall	Precision	F_Score	Accuracy
1	0.745987	0.840426	0.790395	77.7128
2	0.611159	0.757447	0.676485	63.7766
3	0.588975	0.647872	0.617021	59.7872
4	0.693896	0.82234	0.752678	72.9787
5	0.737425	0.779787	0.758014	75.1064
6	0.605405	0.714894	0.65561	62.4468
7	0.690227	0.841489	0.758389	73.1915
8	0.630069	0.681283	0.654676	64.0642
9	0.654528	0.745745	0.697166	67.6064
10	0.635305	0.766486	0.694757	66.3243
11	0.667738	0.669892	0.668814	66.828
12	0.639386	0.797872	0.709891	67.3936

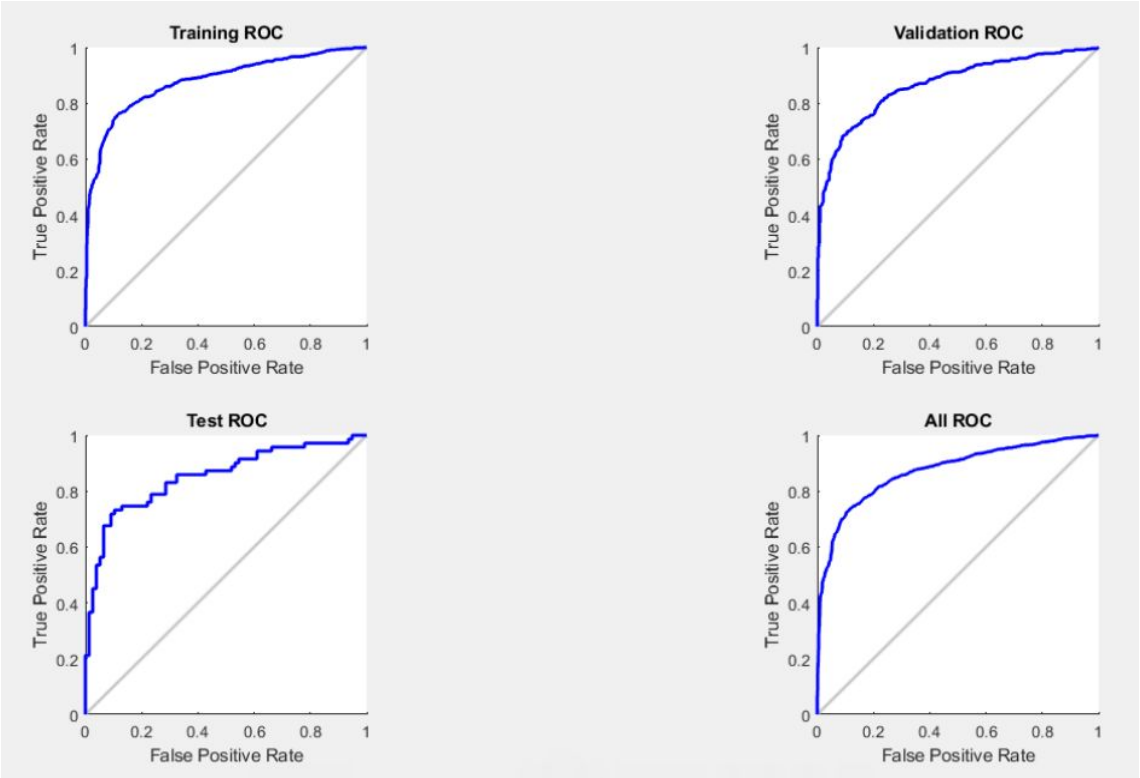
3.4.5.1. PERFORMANCE EVALUATION:



3.4.5.2. CONFUSION MATRIX:



3.4.5.3. Receiver Operating Characteristics:



4. Conclusion

From above results, we conclude that Neural Network outperforms SVM and Decision Tree and this is proved by our performance matrix.

Assignment 2 has much better classification prediction results for Neural Networks while for Assignment 3 have slightly better results for Neural Networks than SVM and Decision Tree.

We have used simple version of Support Vector Machine (SVM) and Decision Tree. Simple SVM works well only when there is linear separability in the two classes while Simple Decision Trees prune a large part of cluster while making any decision. Also Decision Trees are more susceptible to overfitting.

After training the Neural Network for multiple epochs, the model learned better weights which in turn led to optimized results.

We believe that this is because of the complex distribution and high dimensionality of the data. Neural Network has been found to behave better with such data.

5.References:

1. **Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms, Thomas G. Dietterich, tgd@cs.orst.edu, Eun Bae Kong, ebkong@cs.orst.edu, Department of Computer Science, 303 Dearborn Hall, Oregon State University, Corvallis, OR 97331-3202**
2. **Ensemble Methods in Machine Learning, Thomas G. Dietterich, Oregon State University, Corvallis, Oregon, USA, tgd@cs.orst.edu,**
3. **Supervised Machine Learning : A Review of Classification Techniques, S.B. Kotsiantis, Department of Computer Science & Technology, University of Peloponnese, Greece**
4. **Zweig MH, Campbell G (1993) Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. Clinical Chemistry 39:561-577.**