Activity Recognition

CSE 572: Data Mining Assignment 2

TEAM 3

PRATIK BARTAKKE - 1213175715

VIHAR BHATT - 1213139146

VENKATA SAI SANDEEP NADELLA - 1215185621

DARSH PARIKH – 1213185088

Abstract

This project is a part of 2-phase assignment which aims to develop a machine learning model that would be able to recognize eating and non-eating habits. The dataset included the video sequences of several users performing eating actions with fork and spoon while wearing Myo gesture control armband, which tracked the various motions. The Myo Armband included data from accelerometer, gyroscope, orientation and EMG sensors. The dataset is provided along with the eating action labels throughout the video session, while the other actions are labelled as non-eating actions.

In assignment 1, we first cleaned and organized the data, by syncing and merging the ground truth along with sensor data to a common axis. In feature extraction phase, 5 different methods were used to extract features. In the final phase of assignment 1, PCA was implemented on extracted features to reduce the dimensionality of the data.

In this assignment, we would obtain a new set of features by multiplying the PCA output with the feature set. Then 3 different machine learning algorithms were used – Decision Trees, Support Vector Machines (SVM) and Neural Networks. These algorithms were trained on 60% of the dataset, while 40% of the dataset was used for testing purposes. Accuracy metrics like F-1 score, Recall and Precision were used for comparing the results of various models.

Introduction

In this assignment, 3 different machine learning algorithms- Decision Trees, SVM and Neural Network, were utilized to train a classifier to classify actions as either eating action or non-eating action. F-1 score, Precision and Recall metrics were used to compare the performance of the models trained. The assignment is divided into 2 parts

1. **User Dependent Analysis** In this part, set of features for each user for each user were divided int two parts – 1) 60% for training the model 2) 40% for testing the model.
2. **User Independent Analysis** In this part, 60% of total users and all their feature points were considered for training the models. The rest 40% of total users and their corresponding features were used for testing the models trained.

Results

Evaluation Metrics

- **Accuracy**
  Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when we have symmetric datasets where values of false positive and false negatives are almost same. Therefore, we must look at other parameters to evaluate the performance of your model. The problem with using accuracy as main performance metric is that it does do well when you have a severe class imbalance.
  $$\boldsymbol{Accuracy = TP + TN/TP + FP + FN + TN}$$
- **Precision**
  Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.
  $$\boldsymbol{Precision = TP/TP + FP}$$
- **Recall**
  Recall is the ratio of correctly predicted positive observations to the all observations in actual class.
  $$\boldsymbol{Recall = TP/TP + FN}$$
- **F-1 Score**
  F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.
  $$\boldsymbol{F1\ Score = 2 * (Recall * Precision) / (Recall + Precision)}$$

User Dependent Analysis

```
% User dependent analysis
unq = unique(user_eat_matrix);
for u = 1:size(unq,2)
    user = unq(1,u)

    % Get the features corresponding to current user
    feature_user_eat_matrix = [];
    for i = 1:size(user_eat_matrix, 2)
        if user == user_eat_matrix(1,i)
            feature_user_eat_matrix = [feature_user_eat_matrix
new_feature_eat_matrix(:,i)];
```

```
        end
    end

    feature_user_noneat_matrix = [];
    for i = 1:size(user_noneat_matrix, 2)
        if user == user_noneat_matrix(1,i)
            feature_user_noneat_matrix = [feature_user_noneat_matrix
new_feature_noneat_matrix(:,i)];
        end
    end
```

Here we are splitting the train-test sets in 60-40 ratio.

```
% Perform 60% split and use as training data
    feature_user_matrix = [feature_user_eat_matrix feature_user_noneat_matrix];
    Tbl = [feature_user_eat_matrix(:,1:ceil(0.6*size(feature_user_eat_matrix,2)))
feature_user_noneat_matrix(:,1:floor(0.6*size(feature_user_noneat_matrix,2)))]';
    test_data =
[feature_user_eat_matrix(:,ceil(0.6*size(feature_user_eat_matrix,2))+1:end)
feature_user_noneat_matrix(:,floor(0.6*size(feature_user_noneat_matrix,2))+1:end)]'
;

    [size_train,~] = size(Tbl);
    size_test = size(test_data,1);
    train_classes = ones(size_train, 1);
    test_classes = ones(size_test, 1);

    [~, size_train_eat] = size(feature_user_eat_matrix);
    size_train_eat = ceil(size_train_eat*0.6);
    size_test_eat = size(feature_user_eat_matrix,2)-size_train_eat;

    [~,size_train_noneat] = size(feature_user_noneat_matrix);
    size_train_noneat = ceil(size_train_noneat*0.6);
    size_test_noneat = size(feature_user_noneat_matrix,2)-size_train_noneat;
```

User Independent Analysis

```
% User independent analysis
"Overall"
% Perform 60% split and use as training data
feature_matrix = [new_feature_eat_matrix new_feature_noneat_matrix];
Tbl = [new_feature_eat_matrix(:,1:ceil(0.6*size(new_feature_eat_matrix,2)))
new_feature_noneat_matrix(:,1:floor(0.6*size(new_feature_noneat_matrix,2)))]';
test_data =
[new_feature_eat_matrix(:,ceil(0.6*size(new_feature_eat_matrix,2))+1:end)
new_feature_noneat_matrix(:,floor(0.6*size(new_feature_noneat_matrix,2))+1:end)]';

[size_train,~] = size(Tbl);
size_test = size(test_data,1);
train_classes = ones(size_train, 1);
test_classes = ones(size_test, 1);

[~, size_train_eat] = size(new_feature_eat_matrix);
size_train_eat = ceil(size_train_eat*0.6);
size_test_eat = size(new_feature_eat_matrix,2)-size_train_eat;

[~,size_train_noneat] = size(new_feature_noneat_matrix);
size_train_noneat = ceil(size_train_noneat*0.6);
size_test_noneat = size(new_feature_noneat_matrix,2)-size_train_noneat;

train_classes(size_train_eat + 1 : end) = 0;
```

```
test_classes(size_test_eat + 1 : end) = 0;
```

Decision Tree

A Decision Tree is a tree in which the node represents a certain feature, the branches depict a decision rule and each leaf node represents the class [1]. In MATLAB, *fictree* command was used, which utilized the CART Algorithm for training the decision tree classifier.

```
% Decision Tree
    train_classes(size_train_eat + 1 : end) = 0;
    test_classes(size_test_eat + 1 : end) = 0;

    tree = fitctree(Tbl, train_classes);
    labels_dt = predict(tree, test_data);

    figure('Name',['Decision Tree Confusion Matrix for user ' num2str(user)]);
    [conf, order] = confusionmat(test_classes, labels_dt);
    order
    dtf1 = f1Scores(conf)
    plotconfusion(test_classes', labels_dt');
    title(['Decision Tree Confusion Matrix for user ' num2str(user)]);
```

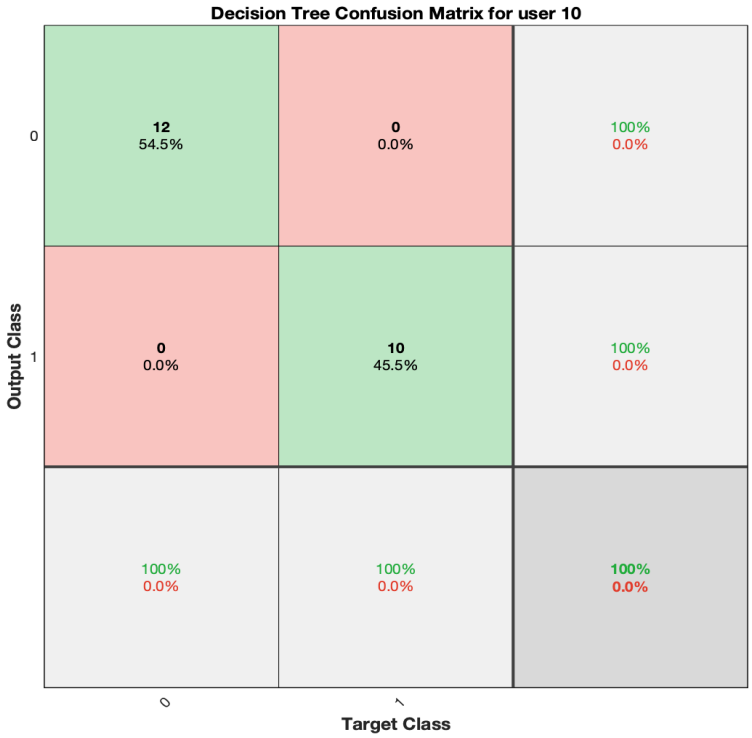
tree = fitctree(Tbl,Y) returns a fitted binary classification decision tree based on the input variables contained in the table Tbl and output in vector Y.

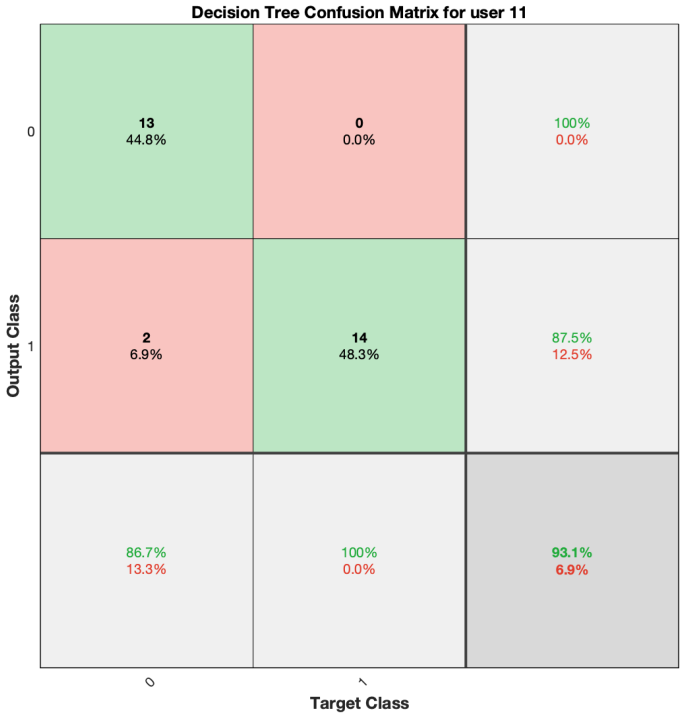
Tbl — Sample data table - Sample data used to train the model, specified as a table. Each row of Tbl corresponds to one observation, and each column corresponds to one predictor variable.

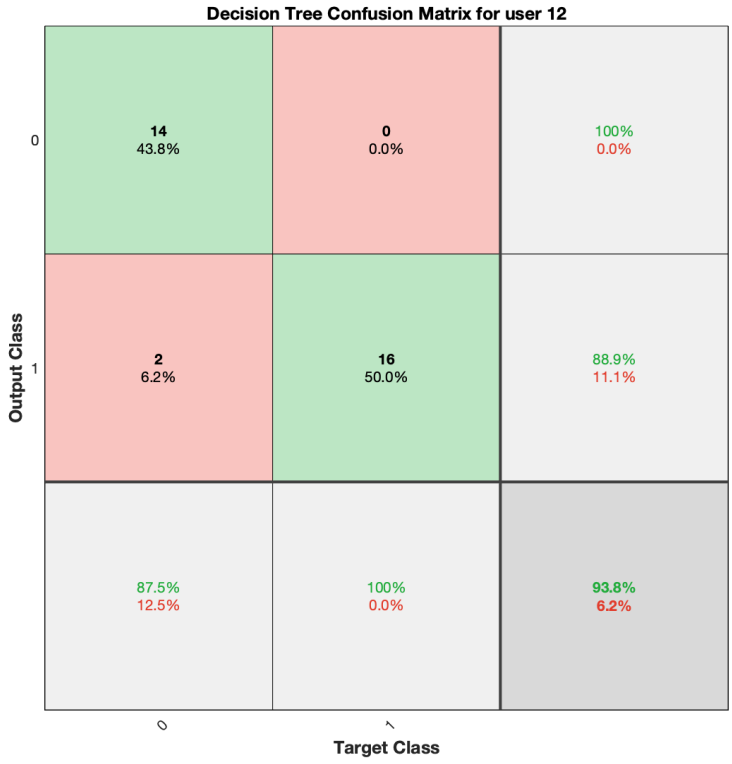
Y — Class labels - Class labels, specified as a numeric vector, categorical vector, logical vector, character array, string array, or cell array of character vectors. Each row of Y represents the classification of the corresponding row of X.

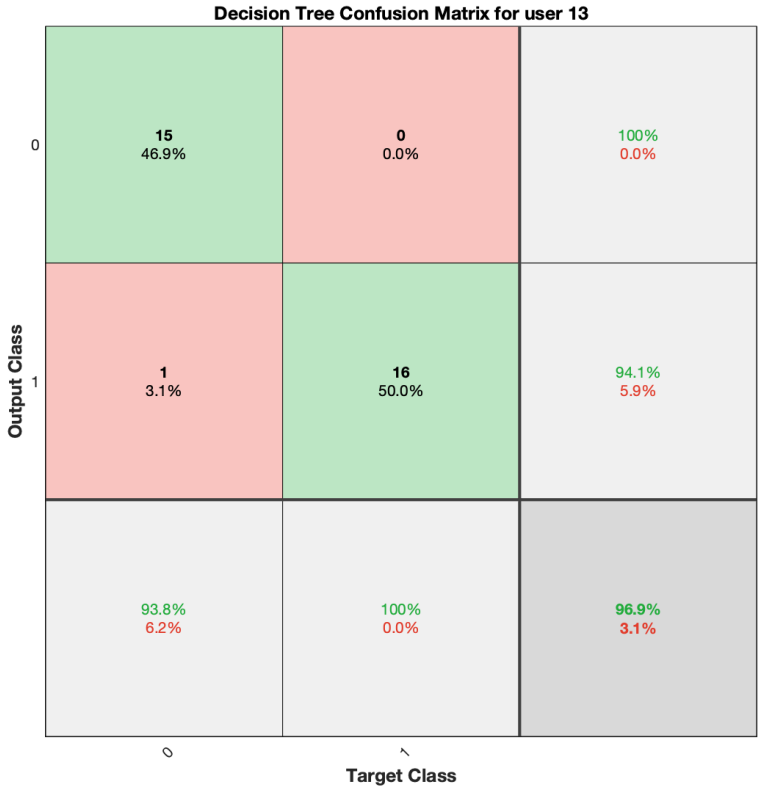
Results for Decision Tree for User dependent analysis

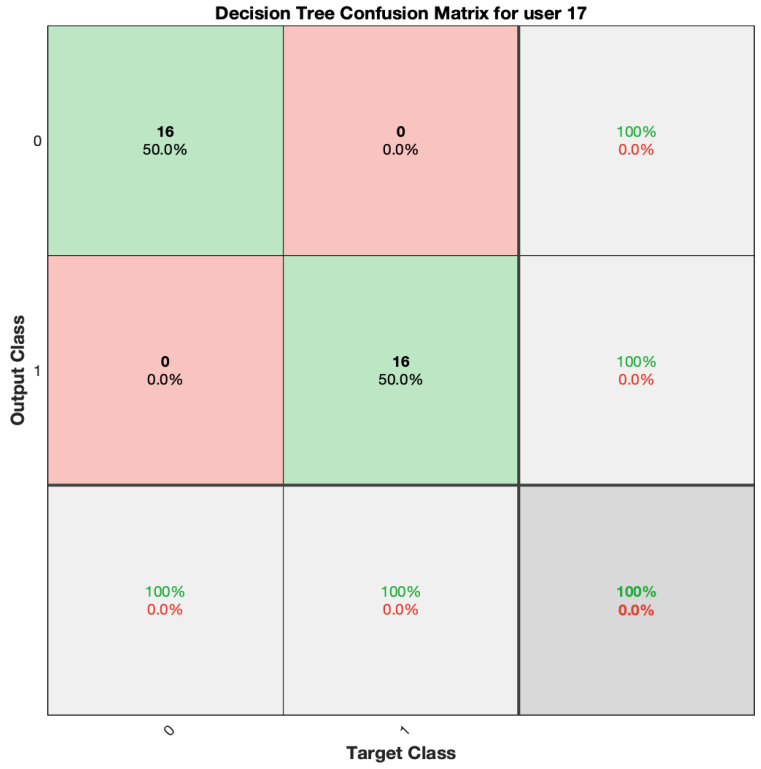
The Confusion Matrix for each user are as follows

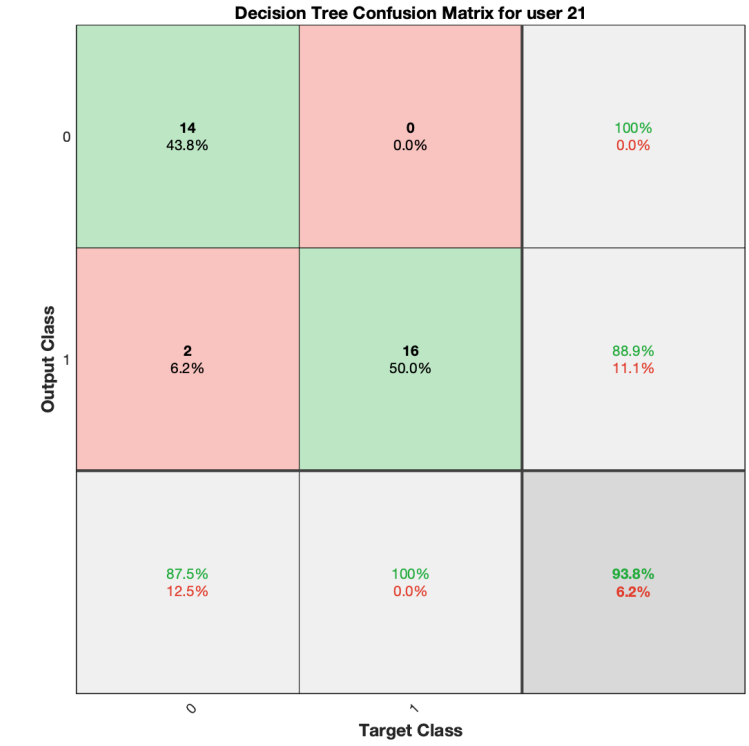


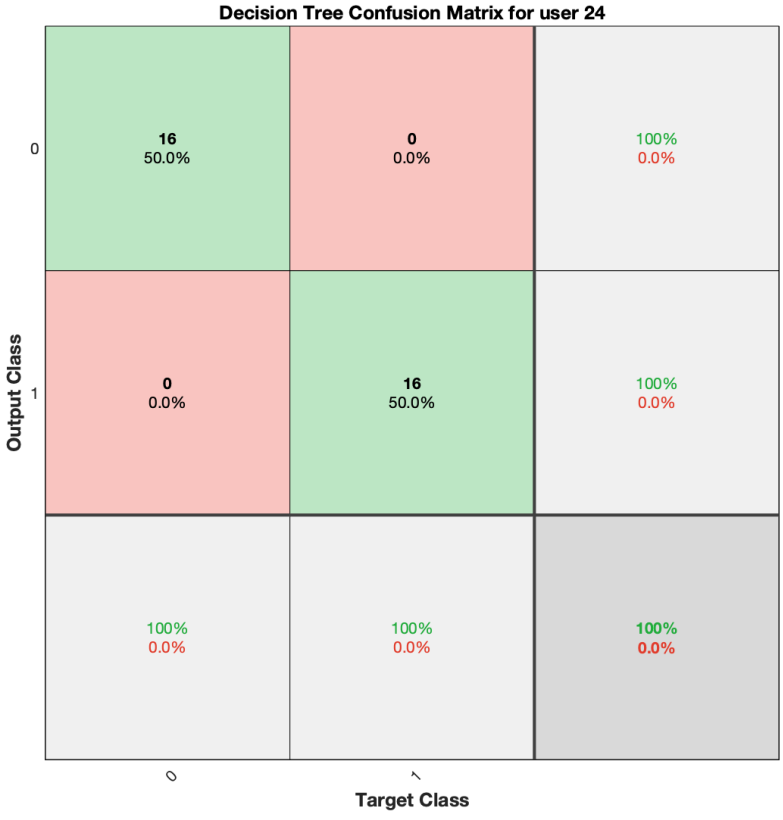

Decision Tree Confusion Matrix for user 11
13
44.8%
0
0.0%
100%
0.0%
2
6.9%
14
48.3%
87.5%
12.5%
86.7%
13.3%
100%
0.0%
93.1%
6.9%
Output Class
Target Class
0
1

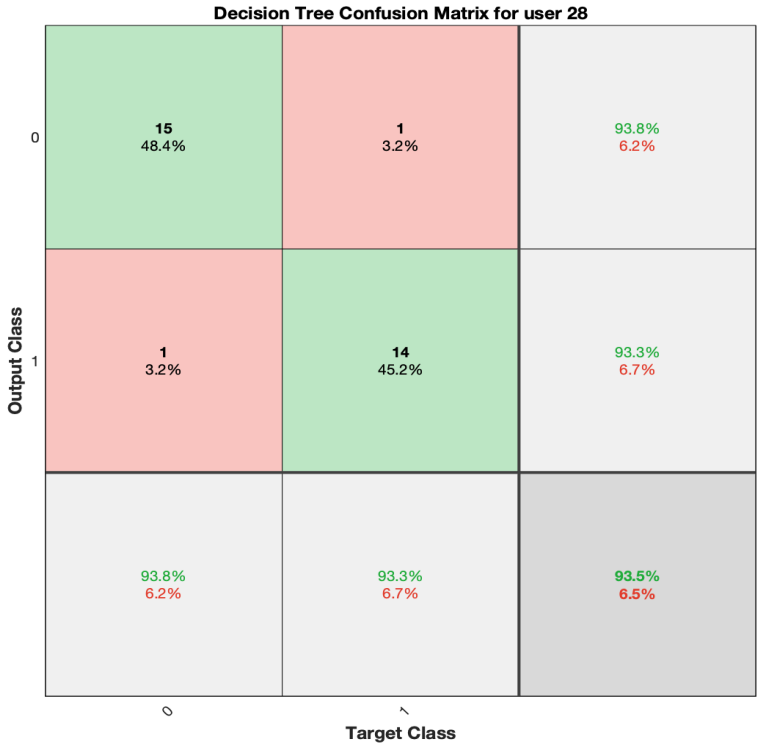

Decision Tree Confusion Matrix for user 12
14
43.8%
0
0.0%
100%
0.0%
2
6.2%
16
50.0%
88.9%
11.1%
87.5%
12.5%
100%
0.0%
93.8%
6.2%
Output Class
Target Class
0
1

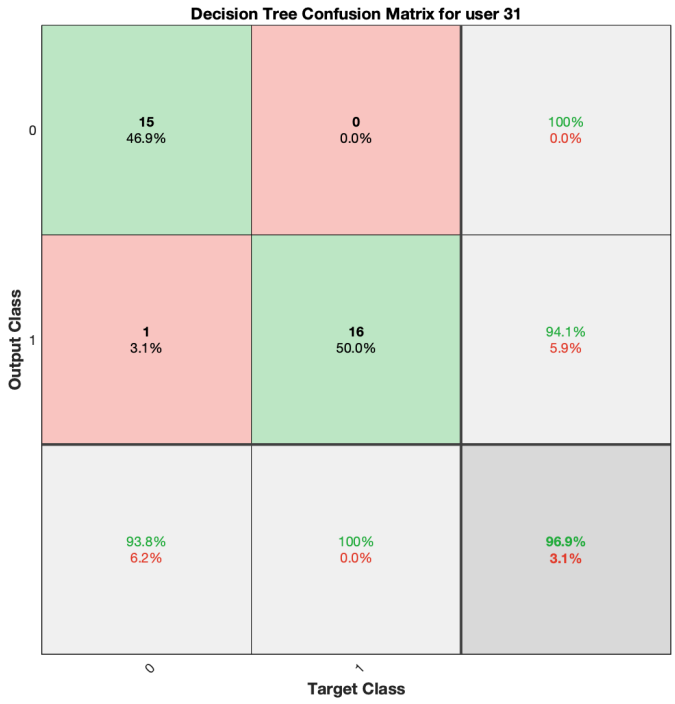

Decision Tree Confusion Matrix for user 13
15
46.9%
0
0.0%
100%
0.0%
1
3.1%
16
50.0%
94.1%
5.9%
93.8%
6.2%
100%
0.0%
96.9%
3.1%
Output Class
0
1
Target Class
0
1

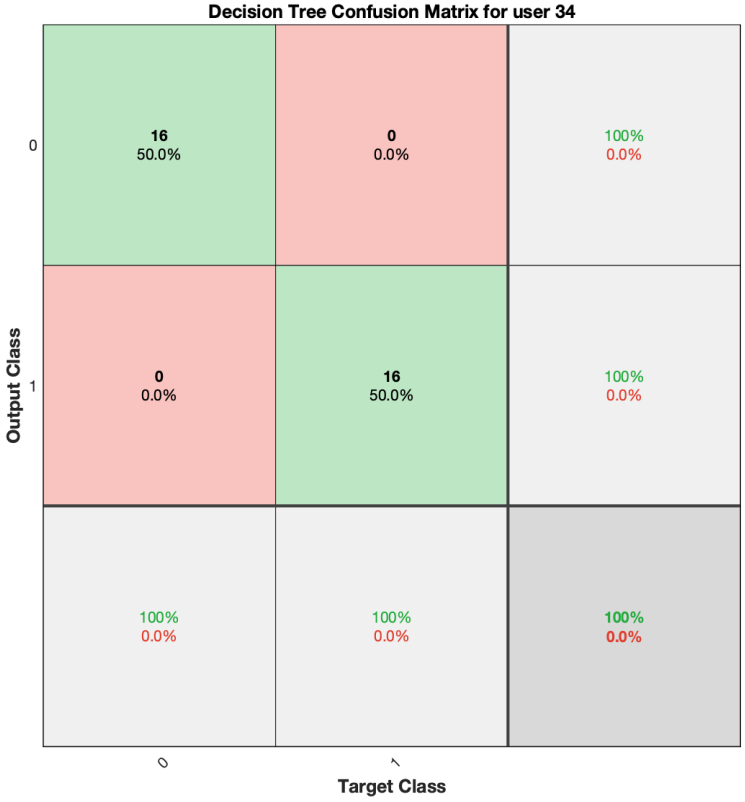

Decision Tree Confusion Matrix for user 17
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
Output Class
0
1
Target Class
0
1

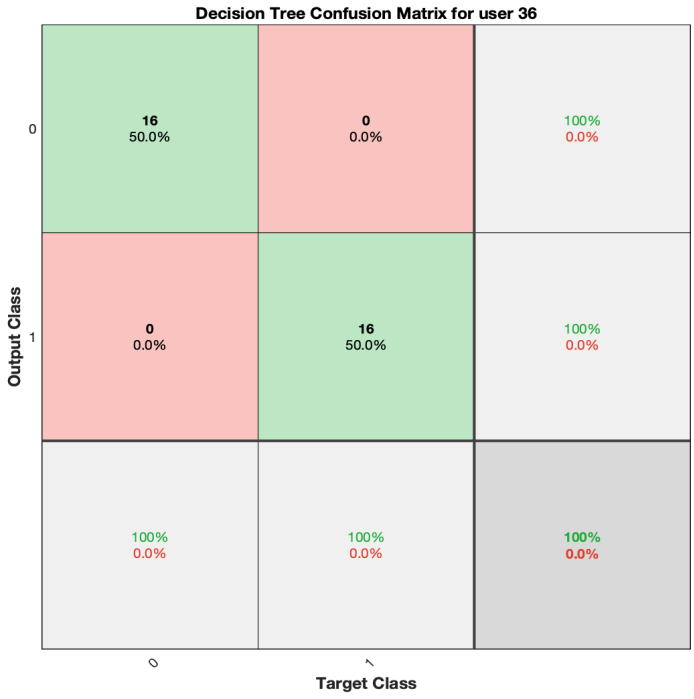

Decision Tree Confusion Matrix for user 21
Output Class
Target Class
0
1
14
43.8%
0
0.0%
100%
0.0%
2
6.2%
16
50.0%
88.9%
11.1%
87.5%
12.5%
100%
0.0%
93.8%
6.2%

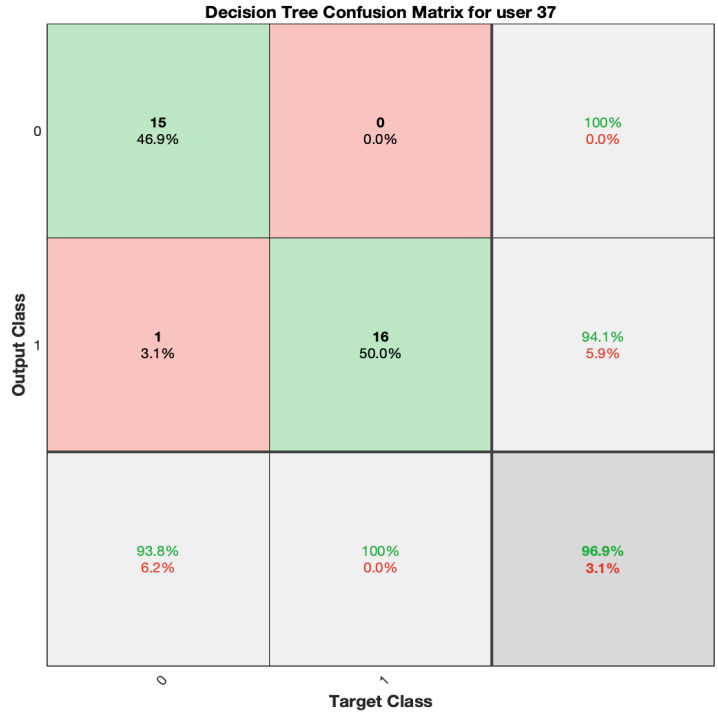

Decision Tree Confusion Matrix for user 24
Output Class
Target Class
0
1
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%

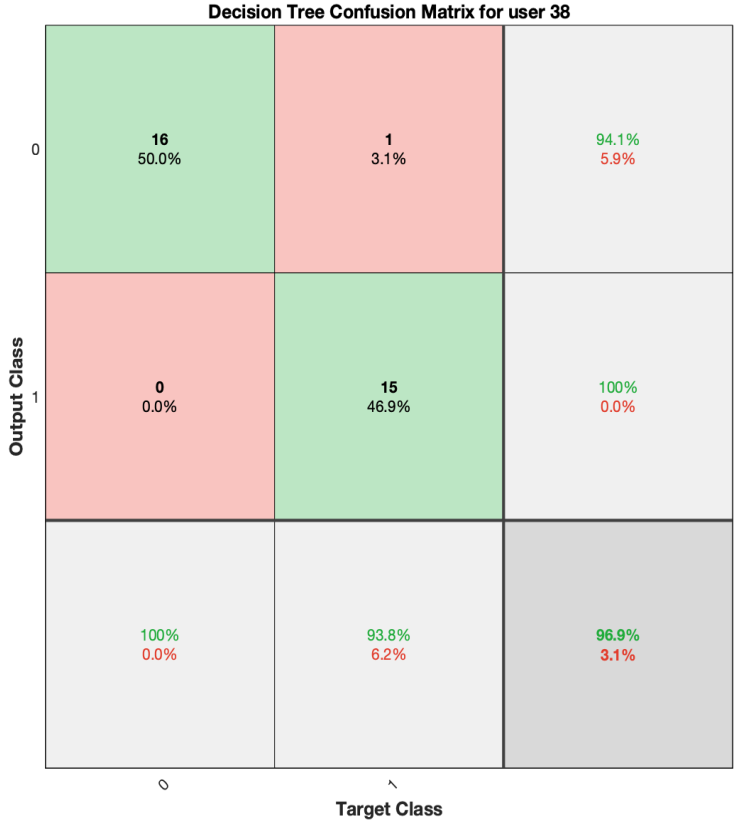

Decision Tree Confusion Matrix for user 28
15
48.4%
1
3.2%
93.8%
6.2%
1
3.2%
14
45.2%
93.3%
6.7%
93.8%
6.2%
93.3%
6.7%
93.5%
6.5%
Output Class
Target Class
0
1

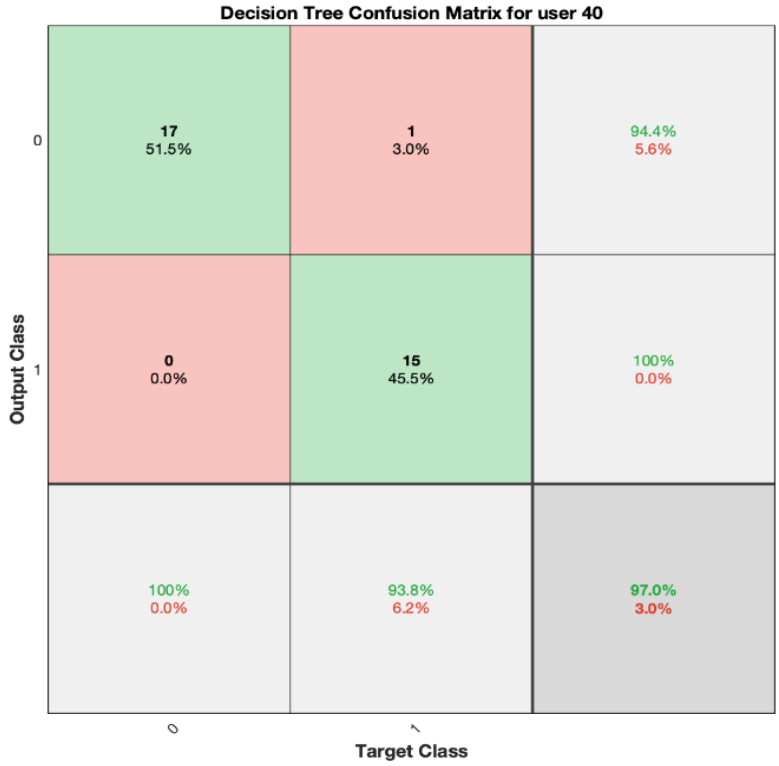

Decision Tree Confusion Matrix for user 31
15
46.9%
0
0.0%
100%
0.0%
1
3.1%
16
50.0%
94.1%
5.9%
93.8%
6.2%
100%
0.0%
96.9%
3.1%
Output Class
Target Class
0
1

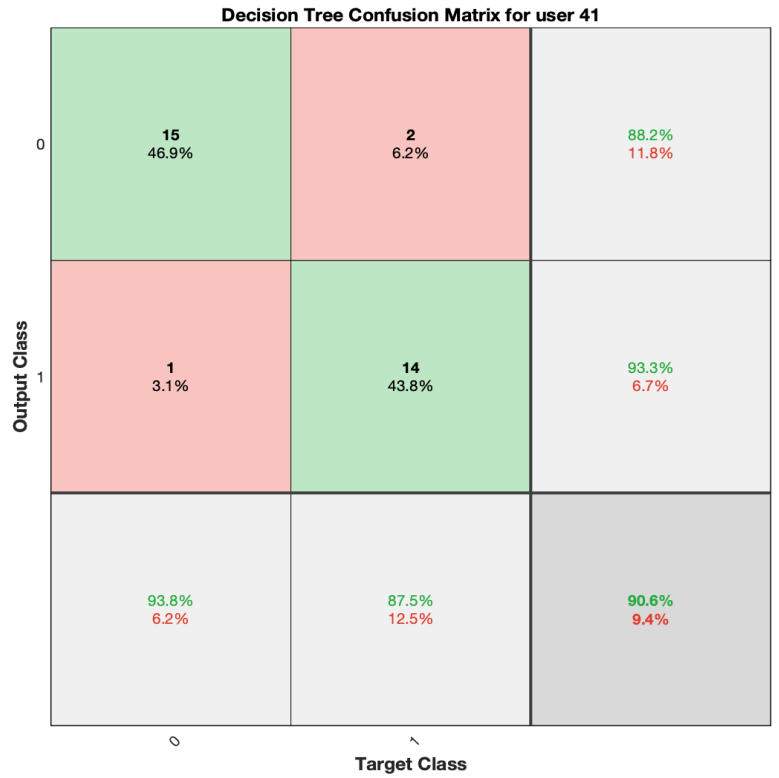

Decision Tree Confusion Matrix for user 34
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class


Decision Tree Confusion Matrix for user 36
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class

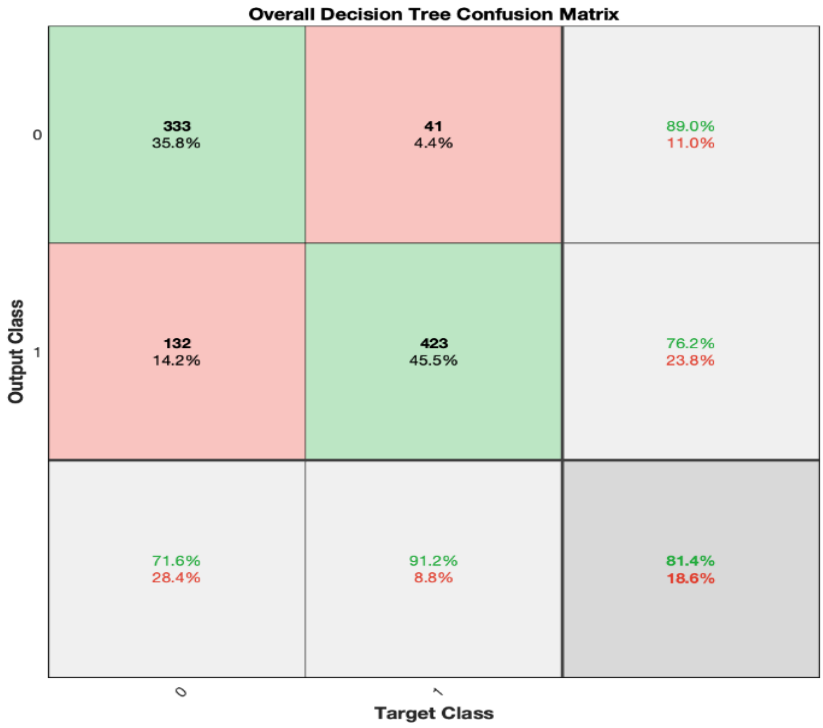

Decision Tree Confusion Matrix for user 37
15
46.9%
0
0.0%
100%
0.0%
1
3.1%
16
50.0%
94.1%
5.9%
93.8%
6.2%
100%
0.0%
96.9%
3.1%
Output Class
Target Class
0
1
0
1


Decision Tree Confusion Matrix for user 38
16
50.0%
1
3.1%
94.1%
5.9%
0
0.0%
15
46.9%
100%
0.0%
100%
0.0%
93.8%
6.2%
96.9%
3.1%
Output Class
Target Class
0
1
0
1


Decision Tree Confusion Matrix for user 40
17
51.5%
1
3.0%
94.4%
5.6%
0
0.0%
15
45.5%
100%
0.0%
100%
0.0%
93.8%
6.2%
97.0%
3.0%
0
1
Output Class
0
1
Target Class


Decision Tree Confusion Matrix for user 41
15
46.9%
2
6.2%
88.2%
11.8%
1
3.1%
14
43.8%
93.3%
6.7%
93.8%
6.2%
87.5%
12.5%
90.6%
9.4%
0
1
Output Class
0
1
Target Class

Results for Decision Tree for User Independent analysis



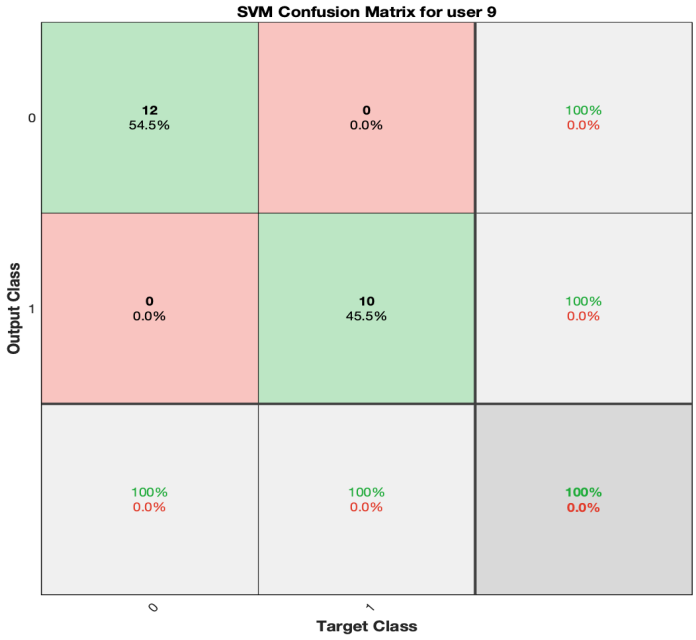
SVM (Support Vector Machine)

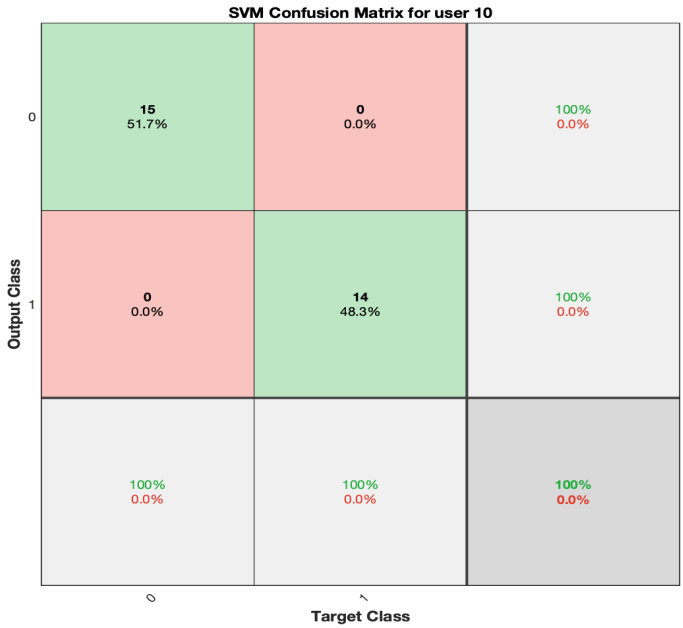
Here we are using the *fitcsvm* function to perform classification for eating and non-eating activities using SVM. The *fitcsvm* trains or cross-validates a support vector machine (SVM) model for two-class (binary) classification on a low-dimensional or moderate-dimensional predictor data set.

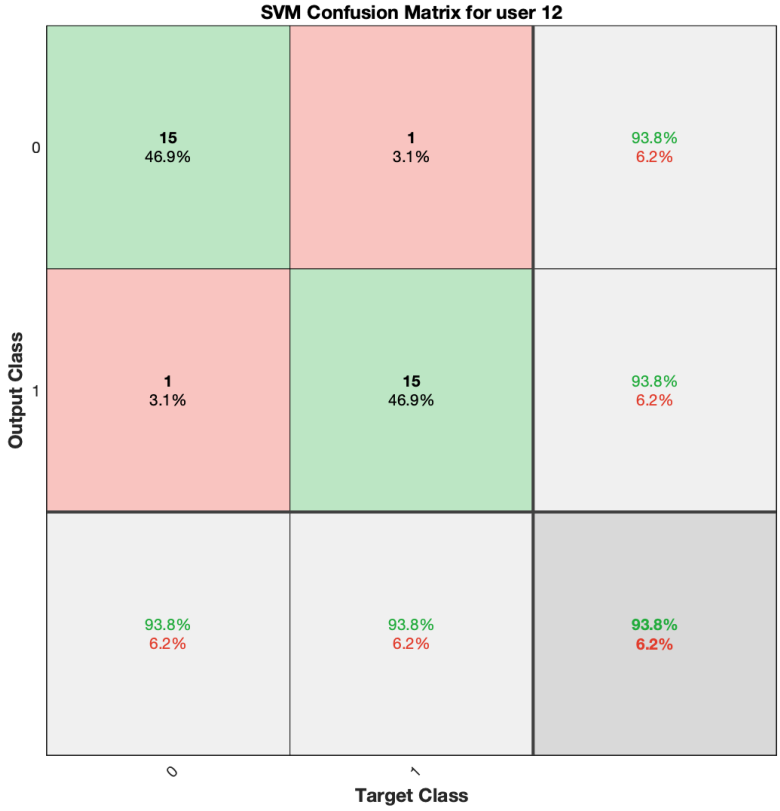
```
% SVM
    train_classes = ones(size_train, 1);
    test_classes = ones(size_test, 1);
    train_classes(size_train_eat + 1 : end) = -1;
    test_classes(size_test_eat + 1 : end) = -1;
    Mdl = fitcsvm(Tbl, train_classes, 'KernelFunction', 'rbf', 'Standardize',
true);
    labels_svm = predict(Mdl, test_data);
    labels_svm = (labels_svm + 1) / 2;
    test_classes_svm = (test_classes + 1) / 2;

    figure('Name',['SVM Confusion Matrix for user ' num2str(user)]);
    [conf, order] = confusionmat(test_classes_svm, labels_svm);
    order
    svmf1 = f1Scores(conf)
    plotconfusion(test_classes_svm', labels_svm');
    title(['SVM Confusion Matrix for user ' num2str(user)]);

    train_classes = ones(size_train, 1);
    test_classes = ones(size_test, 1);
```

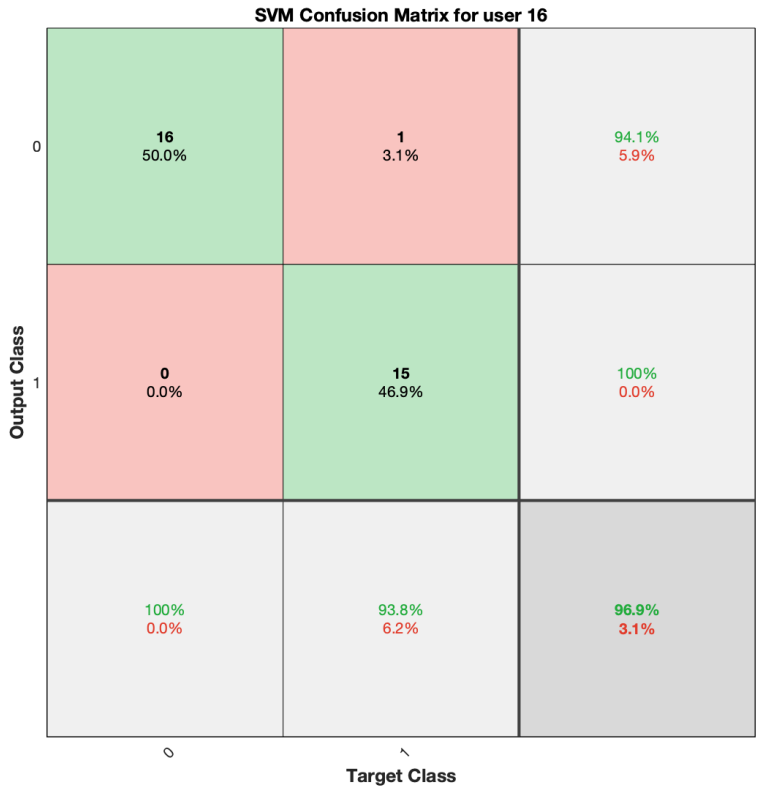
Mdl = fitcsvm(Tbl,Y) returns an SVM classifier trained using the predictor variables in the table Tbl and the class labels in vector Y.

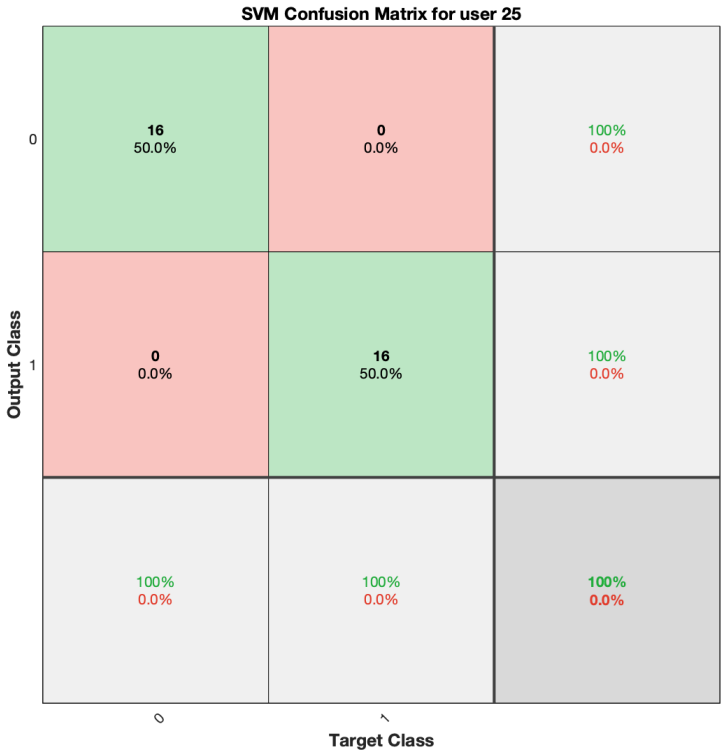
Results for SVM for User Dependent Analysis

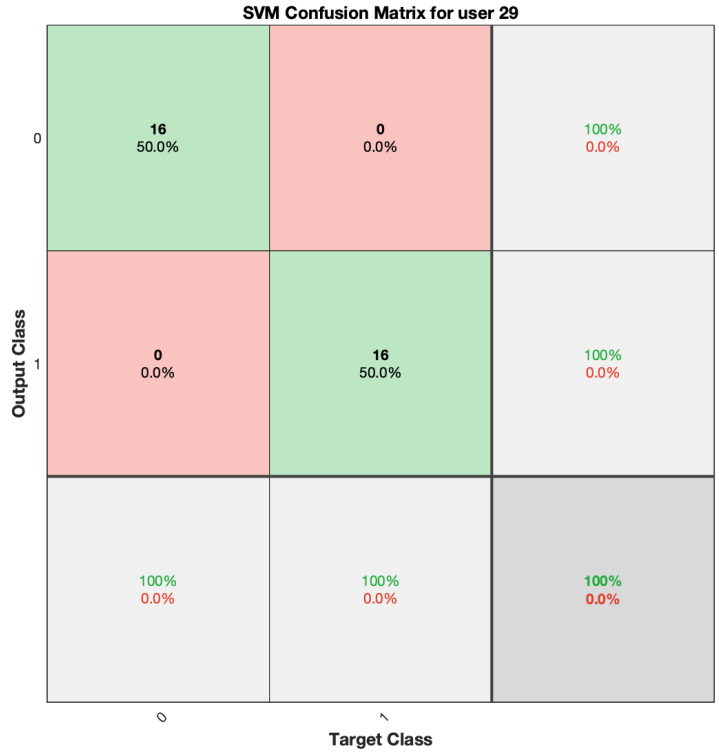


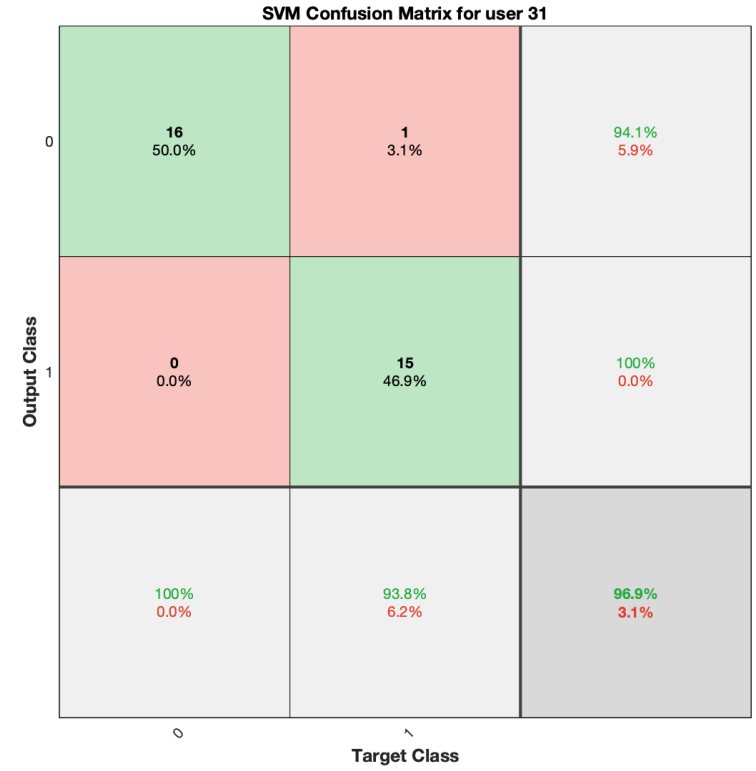


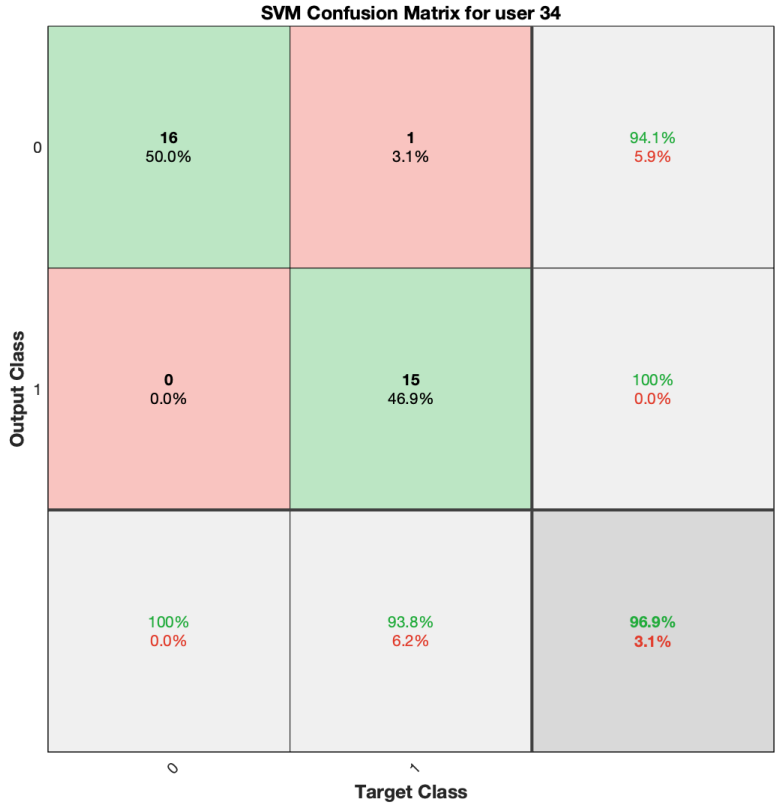

SVM Confusion Matrix for user 12
Output Class
Target Class
0
1
15
46.9%
1
3.1%
93.8%
6.2%
1
3.1%
15
46.9%
93.8%
6.2%
93.8%
6.2%
93.8%
6.2%
93.8%
6.2%

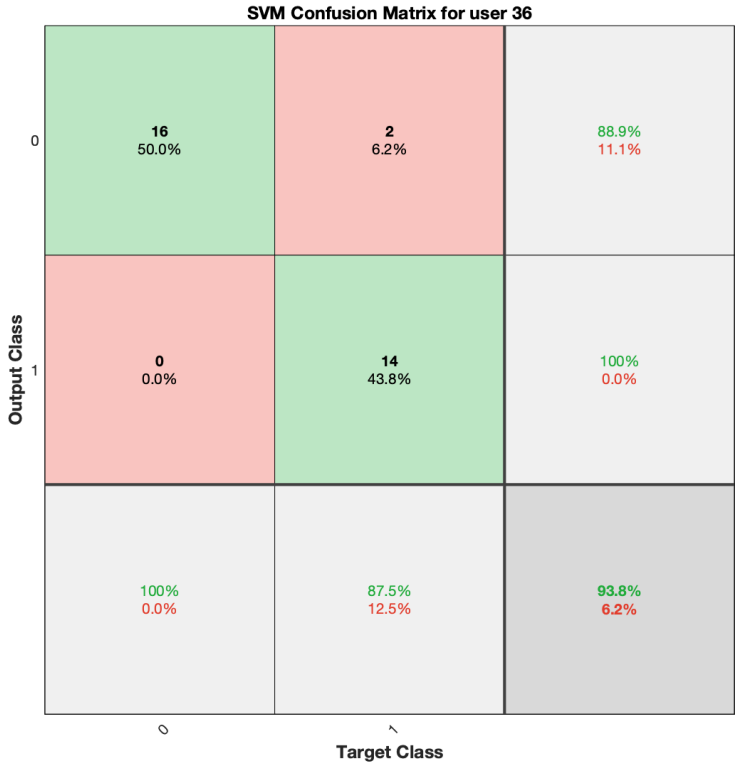

SVM Confusion Matrix for user 16
Output Class
Target Class
0
1
16
50.0%
1
3.1%
94.1%
5.9%
0
0.0%
15
46.9%
100%
0.0%
100%
0.0%
93.8%
6.2%
96.9%
3.1%

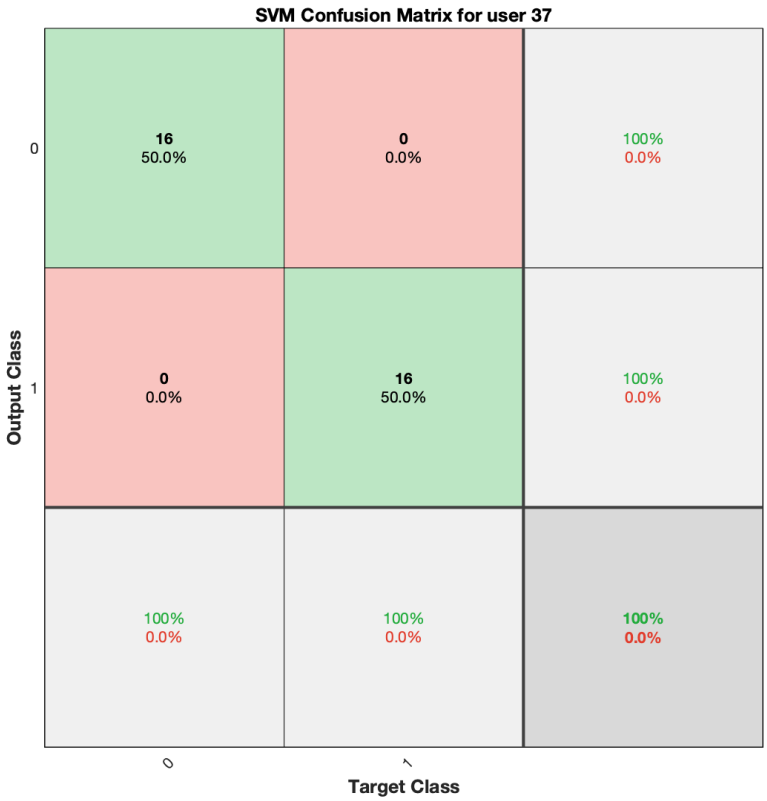

SVM Confusion Matrix for user 25
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class

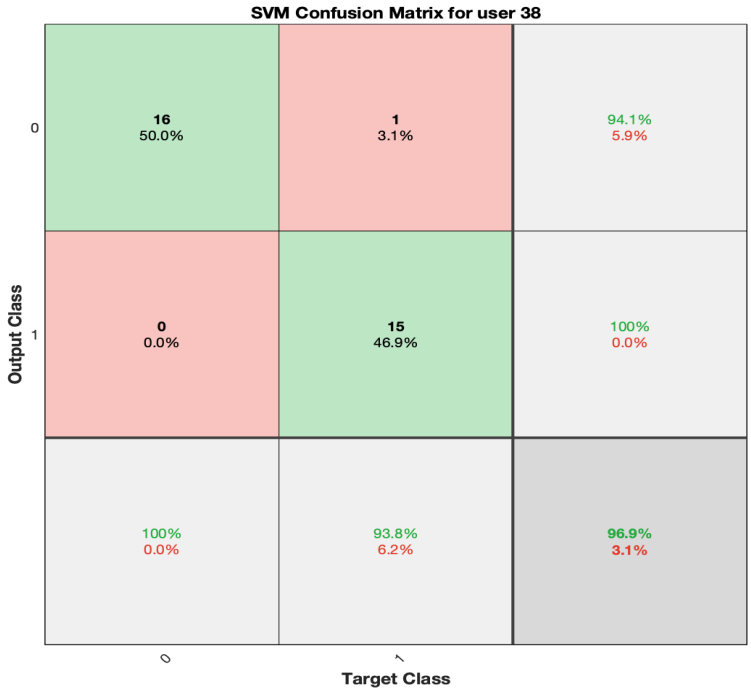

SVM Confusion Matrix for user 29
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class


SVM Confusion Matrix for user 31
16
50.0%
1
3.1%
94.1%
5.9%
0
0.0%
15
46.9%
100%
0.0%
100%
0.0%
93.8%
6.2%
96.9%
3.1%
Output Class
Target Class
0
1

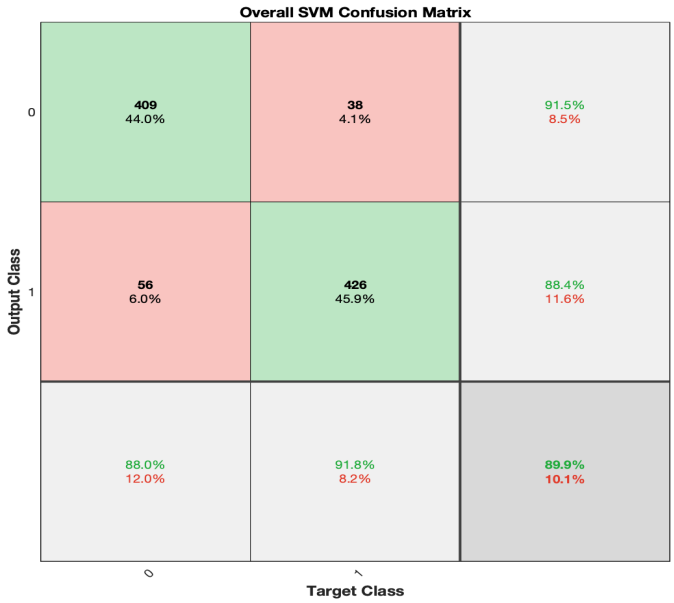

SVM Confusion Matrix for user 34
16
50.0%
1
3.1%
94.1%
5.9%
0
0.0%
15
46.9%
100%
0.0%
100%
0.0%
93.8%
6.2%
96.9%
3.1%
Output Class
Target Class
0
1


SVM Confusion Matrix for user 36
16
50.0%
2
6.2%
88.9%
11.1%
0
0.0%
14
43.8%
100%
0.0%
100%
0.0%
87.5%
12.5%
93.8%
6.2%
0
1
Output Class
0
1
Target Class


SVM Confusion Matrix for user 37
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class



User Independent Analysis



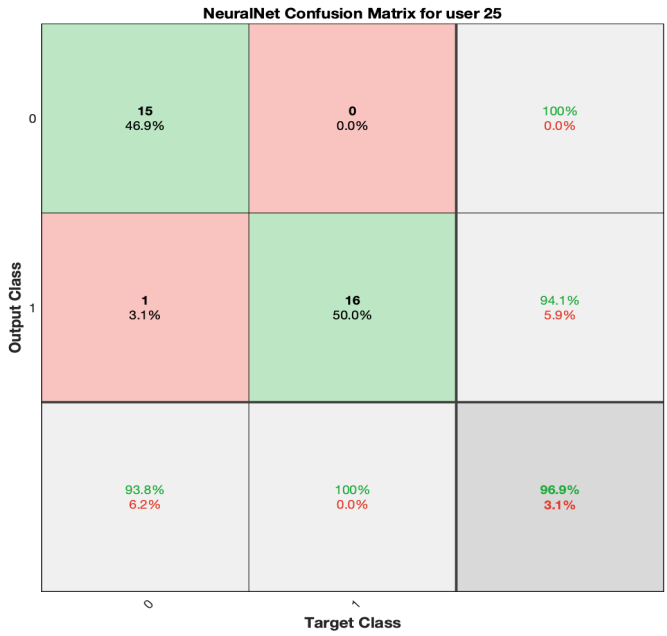
Neural Net

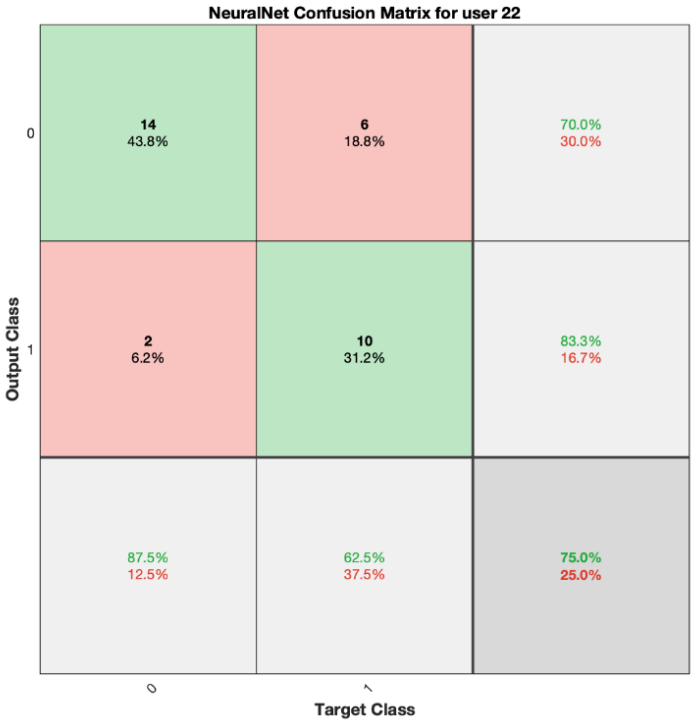
The Deep Learning Toolbox (Also called Neural Network Toolbox) of MATLAB provides a framework for designing, training a neural network for making classifications. For this assignment, we utilized Shallow Neural Network. The configuration of the neural net was as below

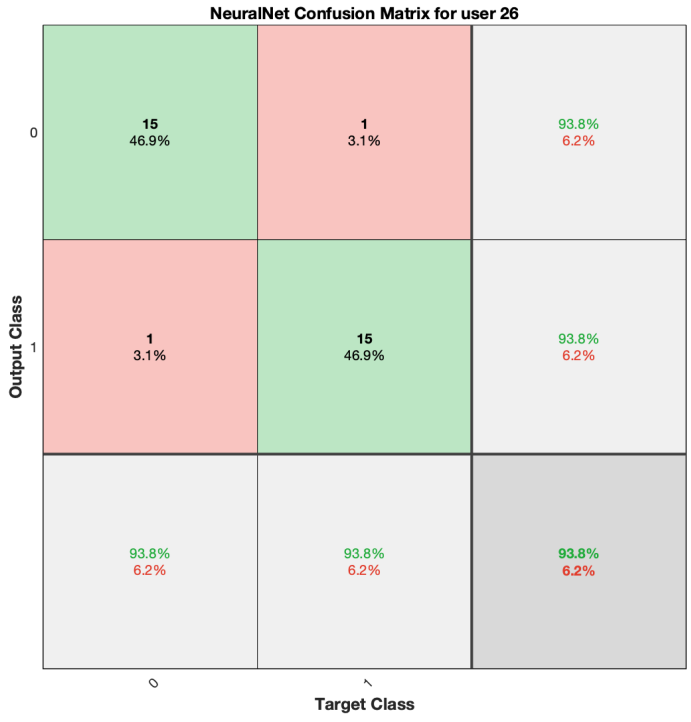
1. Hidden Layers – 10
2. Data Division – Randomly
3. Training with Scaled Conjugate Gradient
4. Minimum Excluded Value used

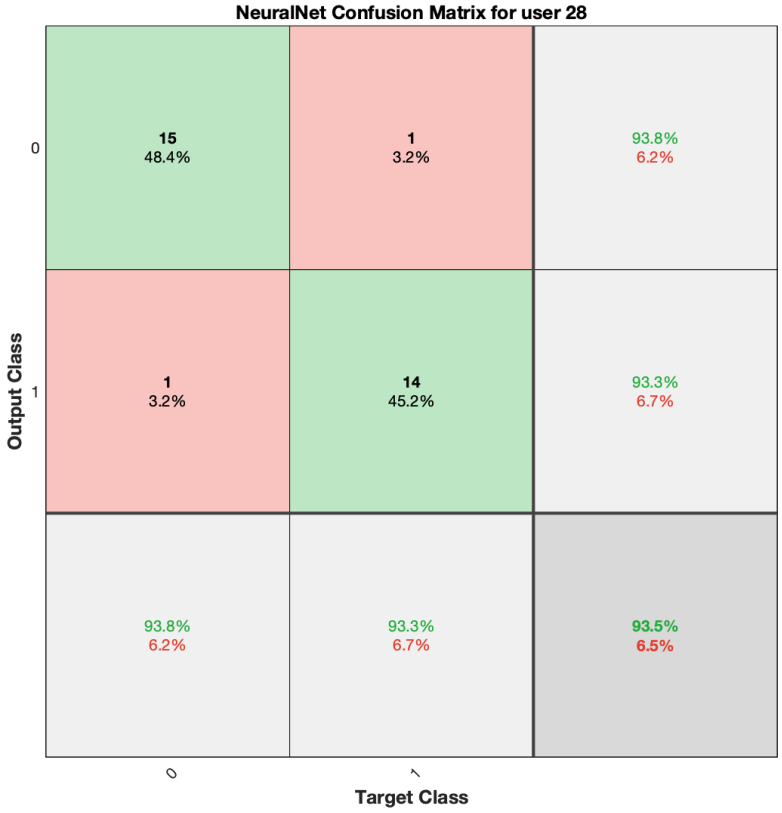
```
% NeuralNet
    train_classes(size_train_eat + 1 : end) = 0;
    test_classes(size_test_eat + 1 : end) = 0;

    net = patternnet(10);
    net.divideFcn = 'divideind';
    net.divideParam.trainInd = 1 : size_train;
    net.divideParam.testInd = size_train + 1 : (size_train + size_test);

    total_training = [Tbl; test_data]';
    total_classes = [train_classes; test_classes]';

    [net, tr] = train(net, total_training, total_classes);
    test_indices = tr.testInd;
    test_outputs = net(total_training(:, test_indices));

    figure('Name',['NeuralNet Confusion Matrix for user ' num2str(user)]);
    [conf, order] = confusionmat(total_classes(:, test_indices)',
round(test_outputs'));
    order
    nnf1 = f1Scores(conf)
    plotconfusion(total_classes(:, test_indices), test_outputs);
    title(['NeuralNet Confusion Matrix for user ' num2str(user)]);
```

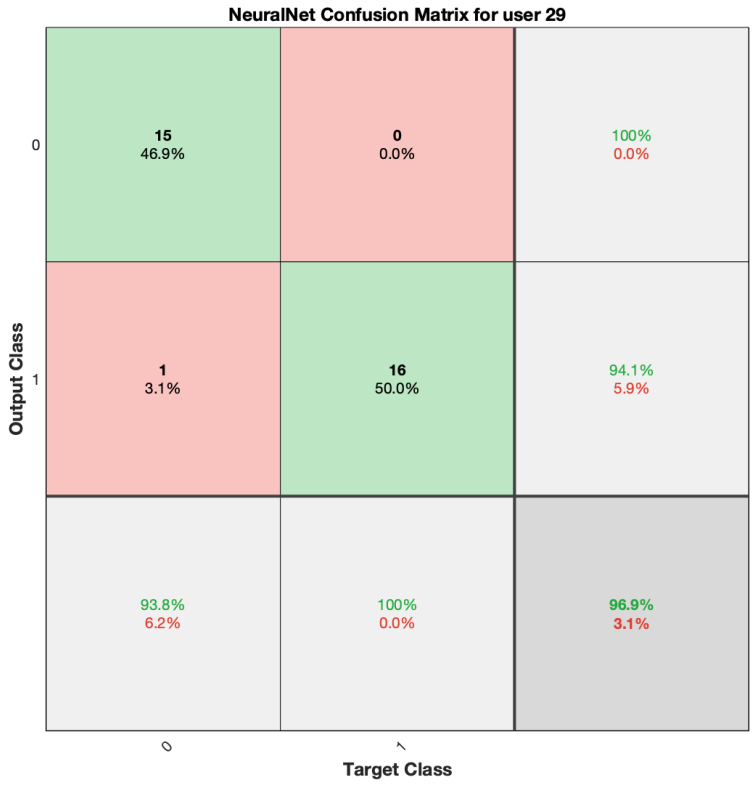
Results for User Dependent Analysis

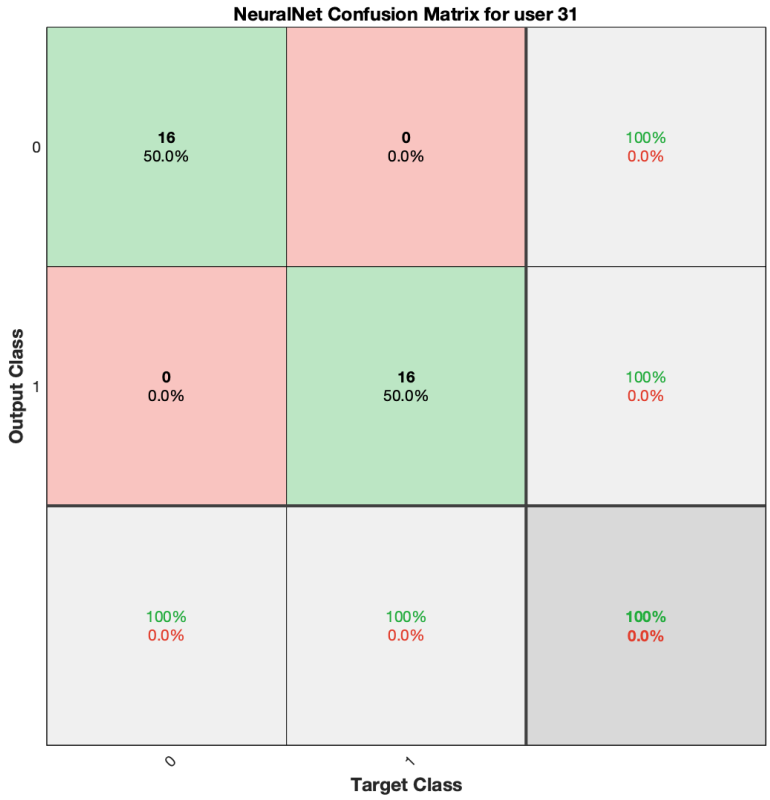


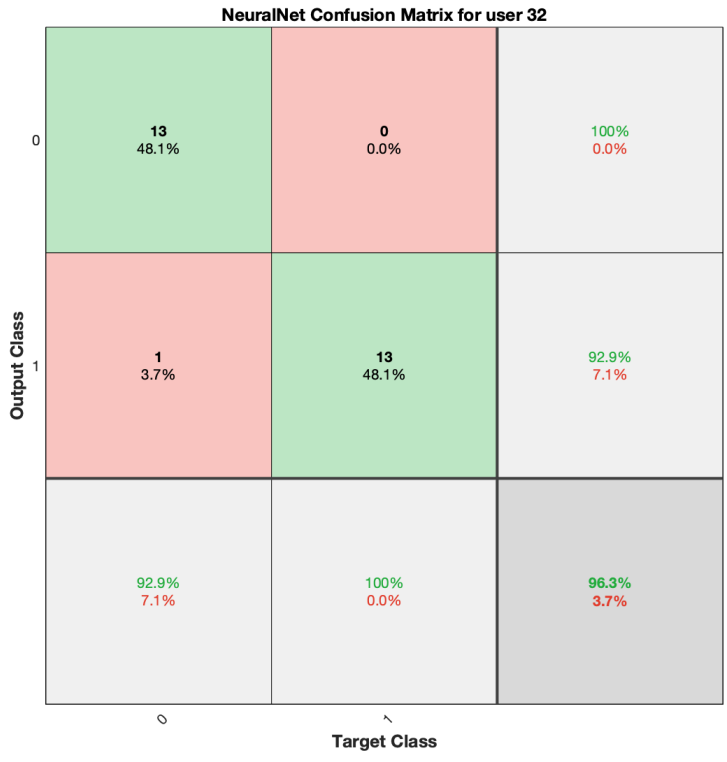

NeuralNet Confusion Matrix for user 22
Output Class
Target Class
0
1
14
43.8%
6
18.8%
70.0%
30.0%
2
6.2%
10
31.2%
83.3%
16.7%
87.5%
12.5%
62.5%
37.5%
75.0%
25.0%

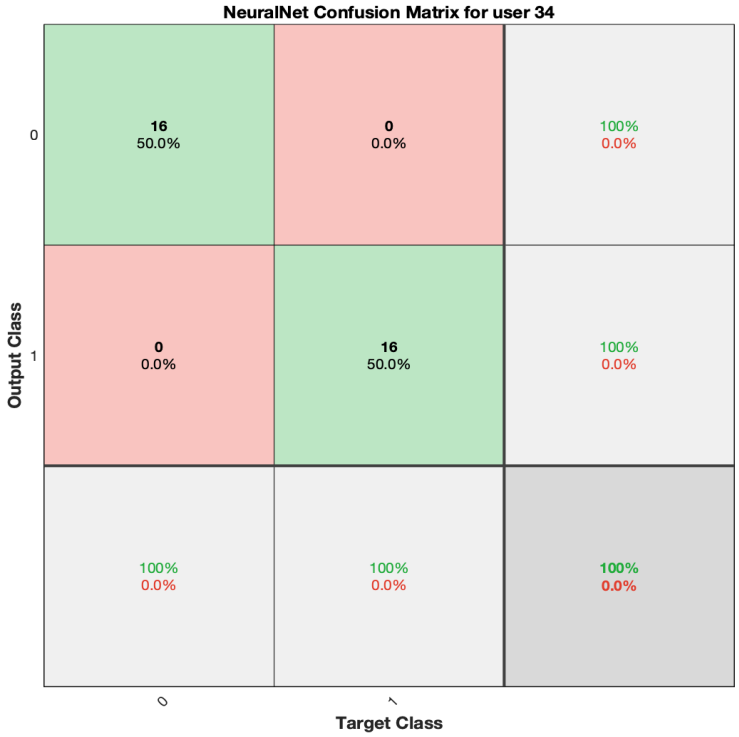

NeuralNet Confusion Matrix for user 26
Output Class
Target Class
0
1
15
46.9%
1
3.1%
93.8%
6.2%
1
3.1%
15
46.9%
93.8%
6.2%
93.8%
6.2%
93.8%
6.2%
93.8%
6.2%

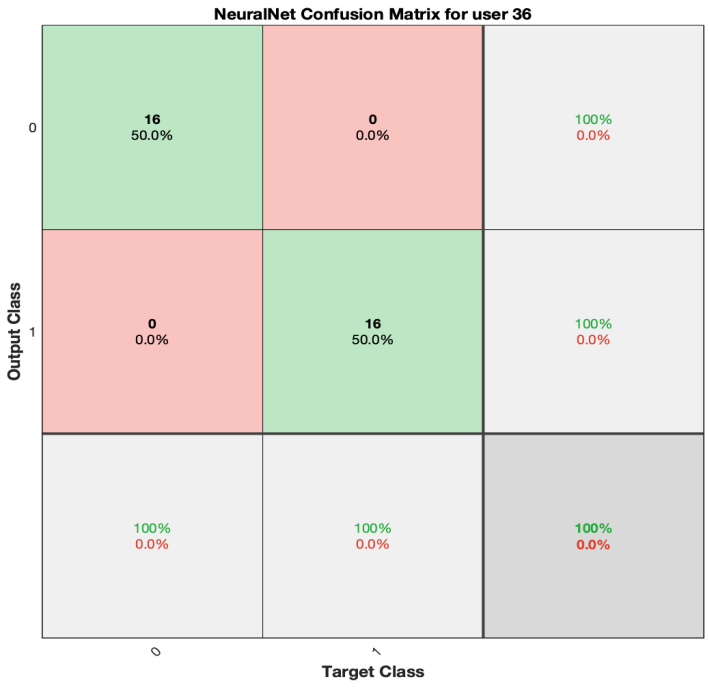

NeuralNet Confusion Matrix for user 28
15
48.4%
1
3.2%
93.8%
6.2%
1
3.2%
14
45.2%
93.3%
6.7%
93.8%
6.2%
93.3%
6.7%
93.5%
6.5%
Output Class
Target Class
0
1
0
1

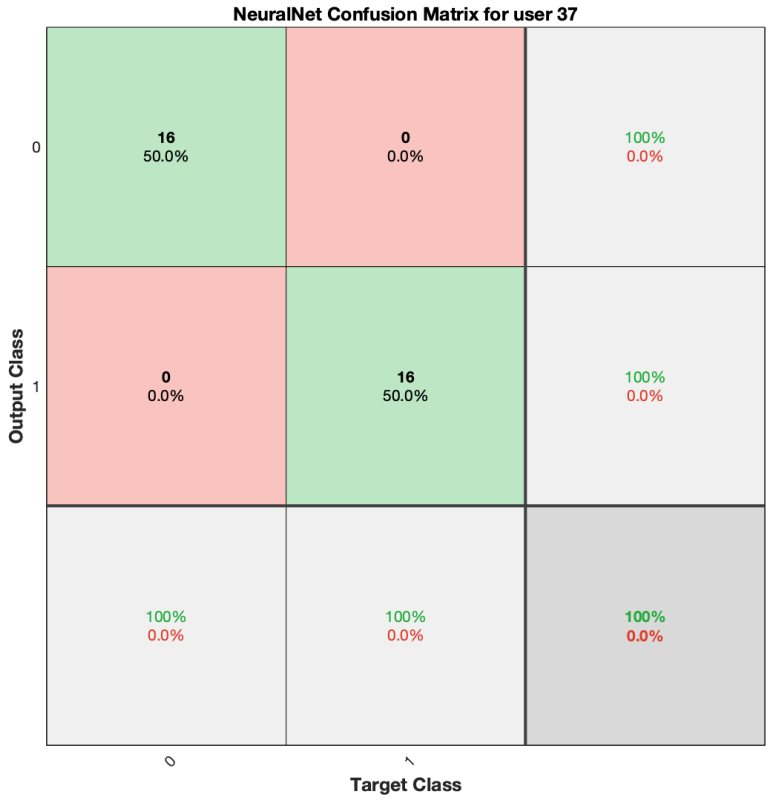

NeuralNet Confusion Matrix for user 29
15
46.9%
0
0.0%
100%
0.0%
1
3.1%
16
50.0%
94.1%
5.9%
93.8%
6.2%
100%
0.0%
96.9%
3.1%
Output Class
Target Class
0
1
0
1

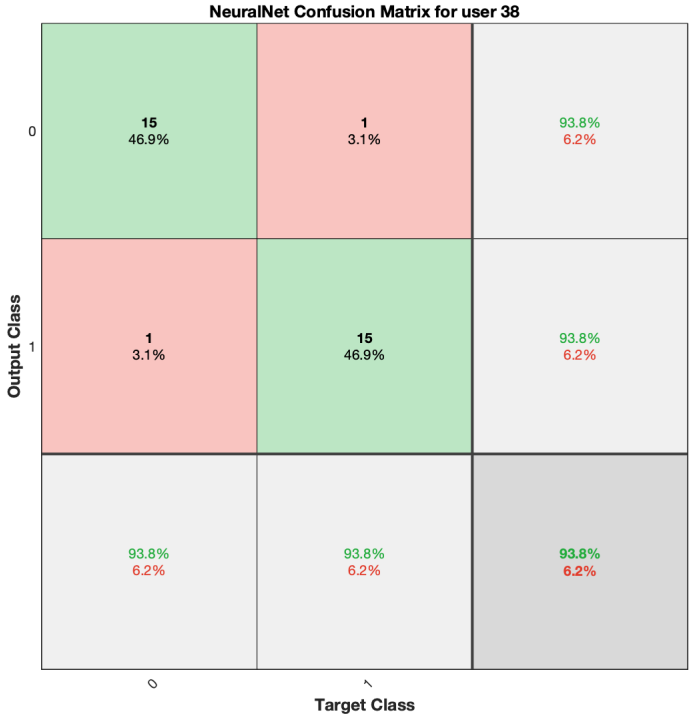

NeuralNet Confusion Matrix for user 31
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class

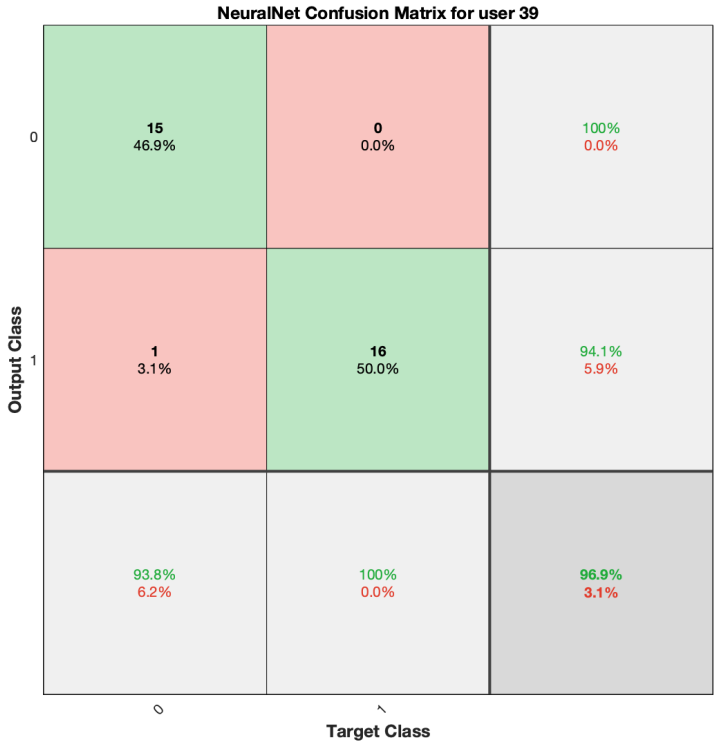

NeuralNet Confusion Matrix for user 32
13
48.1%
0
0.0%
100%
0.0%
1
3.7%
13
48.1%
92.9%
7.1%
92.9%
7.1%
100%
0.0%
96.3%
3.7%
0
1
Output Class
0
1
Target Class

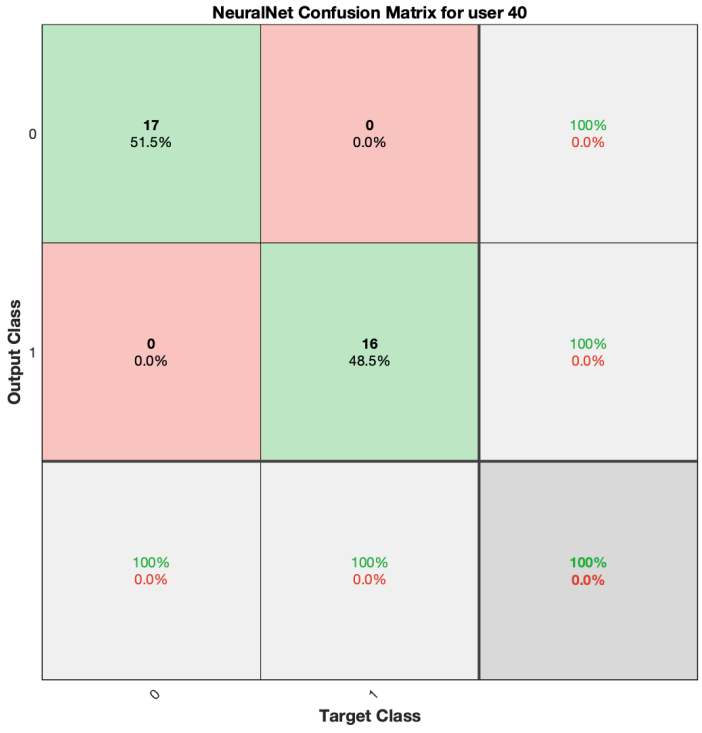

NeuralNet Confusion Matrix for user 34
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class

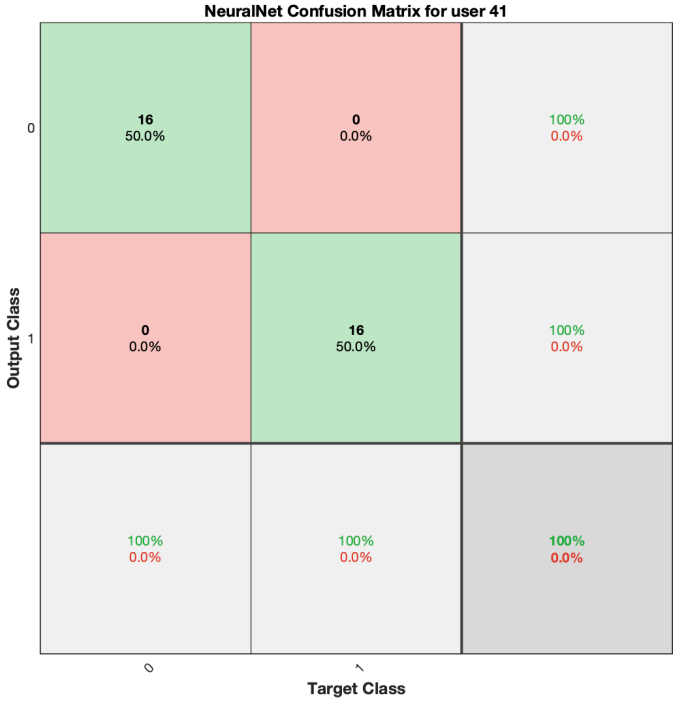

NeuralNet Confusion Matrix for user 36
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class


NeuralNet Confusion Matrix for user 37
16
50.0%
0
0.0%
100%
0.0%
0
0.0%
16
50.0%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
0
1
Output Class
0
1
Target Class

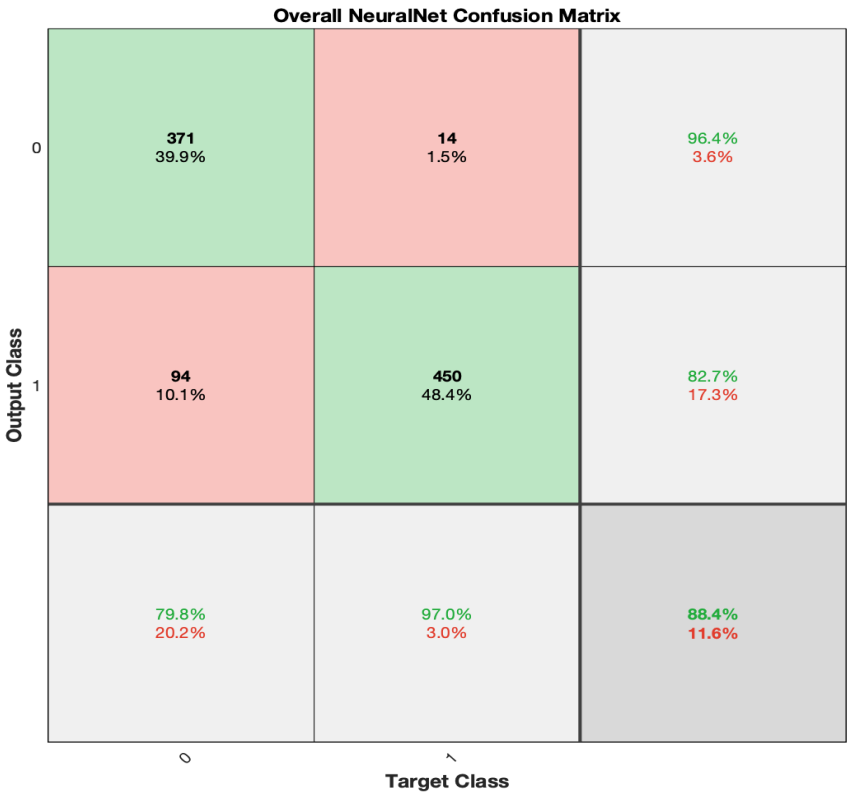

NeuralNet Confusion Matrix for user 38
15
46.9%
1
3.1%
93.8%
6.2%
1
3.1%
15
46.9%
93.8%
6.2%
93.8%
6.2%
93.8%
6.2%
93.8%
6.2%
0
1
Output Class
0
1
Target Class


NeuralNet Confusion Matrix for user 39
15
46.9%
0
0.0%
100%
0.0%
1
3.1%
16
50.0%
94.1%
5.9%
93.8%
6.2%
100%
0.0%
96.9%
3.1%
Output Class
0
1
Target Class
0
1


NeuralNet Confusion Matrix for user 40
17
51.5%
0
0.0%
100%
0.0%
0
0.0%
16
48.5%
100%
0.0%
100%
0.0%
100%
0.0%
100%
0.0%
Output Class
0
1
Target Class
0
1



Results for User Independent Analysis



Evaluation Metrics for Classifiers

Decision Tree

User Dependent Analysis

Table showing the statistics for evaluation metrics for each user for **Eating Activity**

User	Accuracy	Precision	Recall	F-1
9	90.9	90	90	90
10	100	100	100	100
11	93.1	87.5	100	93.33333
12	93.8	88.9	100	94.12388
13	96.9	94.1	100	96.96033
14	65.2	61.5	72.7	66.63264
16	75	83.3	62.5	71.41632
17	100	100	100	100
18	84.4	82.6	87.5	84.97942
19	55.6	53.3	61.5	57.10714
21	93.8	88.9	100	94.12388
22	59.4	58.8	62.5	60.59357
23	87.5	80	100	88.88889
24	100	100	100	100
25	71.9	66.7	87.5	75.69715
28	93.5	93.3	93.3	93.3
30	68.8	61.5	100	76.16099
31	96.9	94.1	100	96.96033
32	81.5	72.2	100	83.85598
34	100	100	100	100
36	100	100	100	100
37	96.9	94.1	100	96.96033
38	96.9	100	93.8	96.80083
39	90.6	93.3	87.5	90.30697
40	97	100	93.8	96.80083
41	90.6	93.3	87.5	90.30697

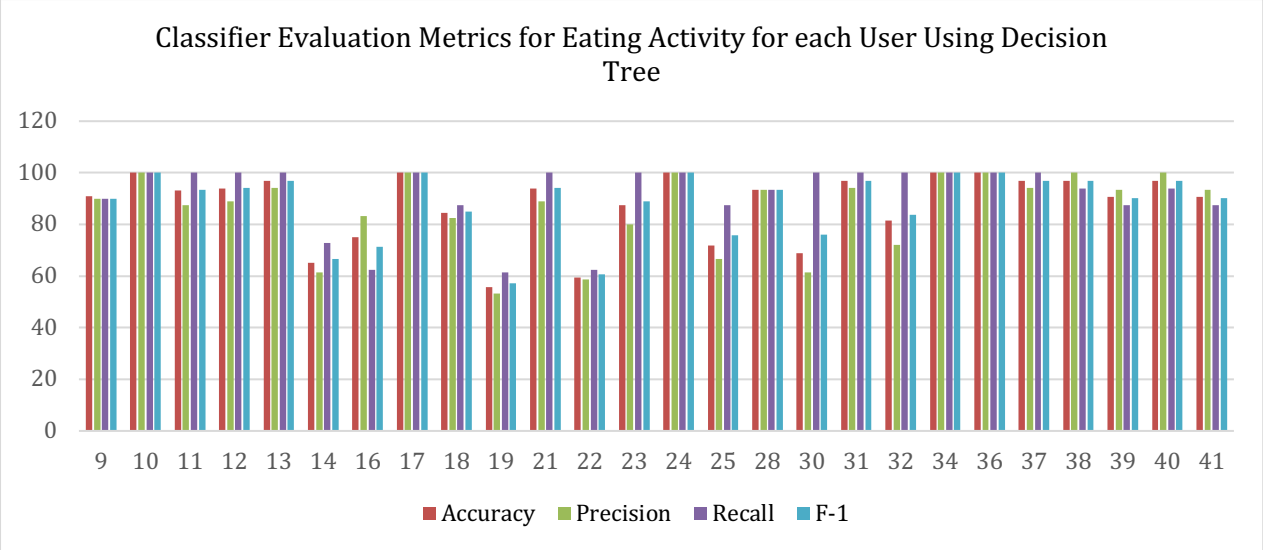
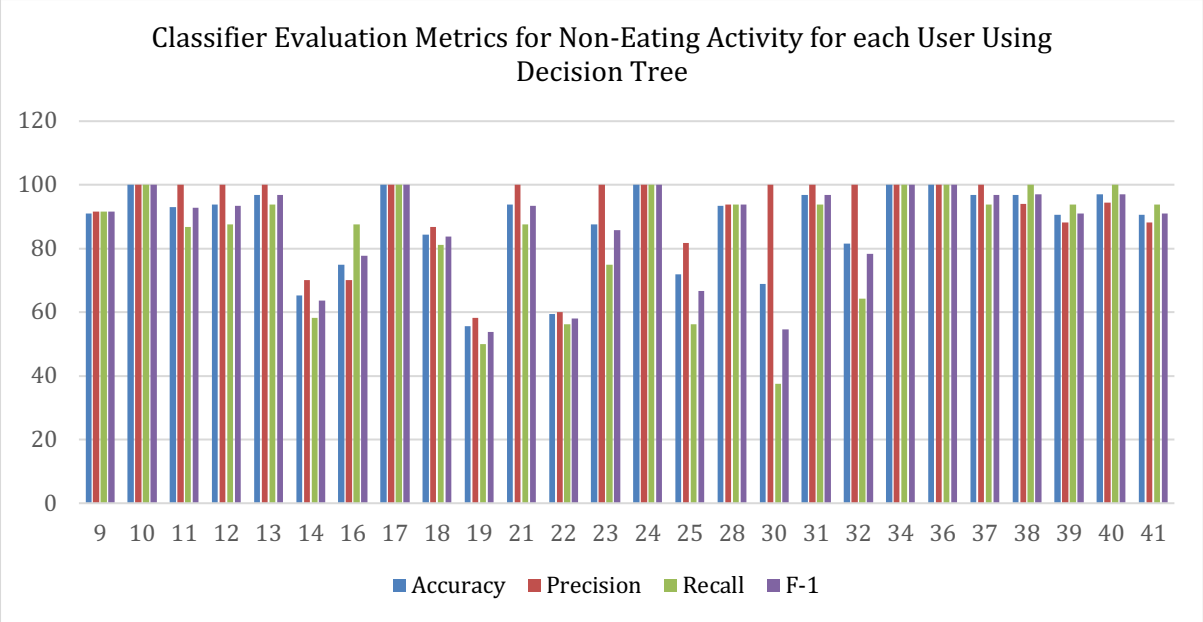


Table showing the statistics for evaluation metrics for each user for **Non-Eating Activity**

User	Accuracy	Precision	Recall	F-1
9	90.9	91.7	91.7	91.7
10	100	100	100	100
11	93.1	100	86.7	92.87627
12	93.8	100	87.5	93.33333
13	96.9	100	93.8	96.80083
14	65.2	70	58.3	63.61652
16	75	70	87.5	77.77778
17	100	100	100	100
18	84.4	86.7	81.2	83.85992
19	55.6	58.3	50	53.83195
21	93.8	100	87.5	93.33333
22	59.4	60	56.2	58.03787
23	87.5	100	75	85.71429
24	100	100	100	100
25	71.9	81.8	56.2	66.62551
28	93.5	93.8	93.8	93.8
30	68.8	100	37.5	54.54545
31	96.9	100	93.8	96.80083
32	81.5	100	64.3	78.27145
34	100	100	100	100
36	100	100	100	100
37	96.9	100	93.8	96.80083
38	96.9	94.1	100	96.96033
39	90.6	88.2	93.8	90.91385
40	97	94.4	100	97.11934
41	90.6	88.2	93.8	90.91385



SVM

User Dependent Analysis

Table showing the statistics for evaluation metrics for each user for **Eating Activity**

User	Accuracy	Precision	Recall	F-1
9	100	100	100	100
10	100	100	100	100
11	87.5	100	75	85.7142857
12	93.8	93.8	93.8	93.8
13	73.9	69.2	81.8	74.9743046
14	71.9	76.9	62.5	68.956241
16	96.9	78.6	68.8	73.3742198
17	75	78.6	68.8	73.3742198
18	51.9	50	53.8	51.8304432
19	78.1	84.6	68.8	75.8863103
21	56.2	55	68.8	61.1308562
22	71.9	81.8	56.2	66.6255072
23	90.6	100	81.2	89.6247241
24	87.5	87.5	87.5	87.5
25	100	100	100	100
26	84.4	76.2	100	86.492622
27	90.6	93.3	87.5	90.306969
28	87.1	92.3	80	85.7109692
29	100	100	100	100
30	53.1	55.6	31.2	39.9705069
31	96.9	100	93.8	96.8008256
32	70.4	85.7	46.2	60.0354814

33	78.1	80	75	77.4193548
34	96.9	100	93.8	96.8008256
36	93.8	100	87.5	93.3333333
37	100	100	100	100
38	96.9	100	93.8	96.8008256
39	87.5	100	75	85.7142857
40	87.9	100	75	85.7142857
41	84.4	100	68.8	81.5165877

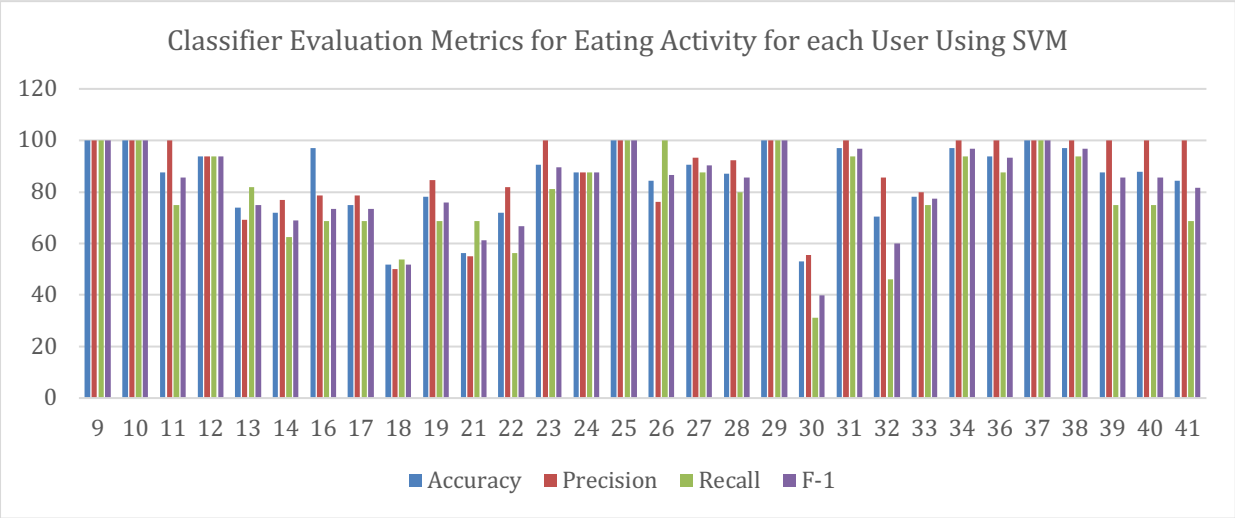
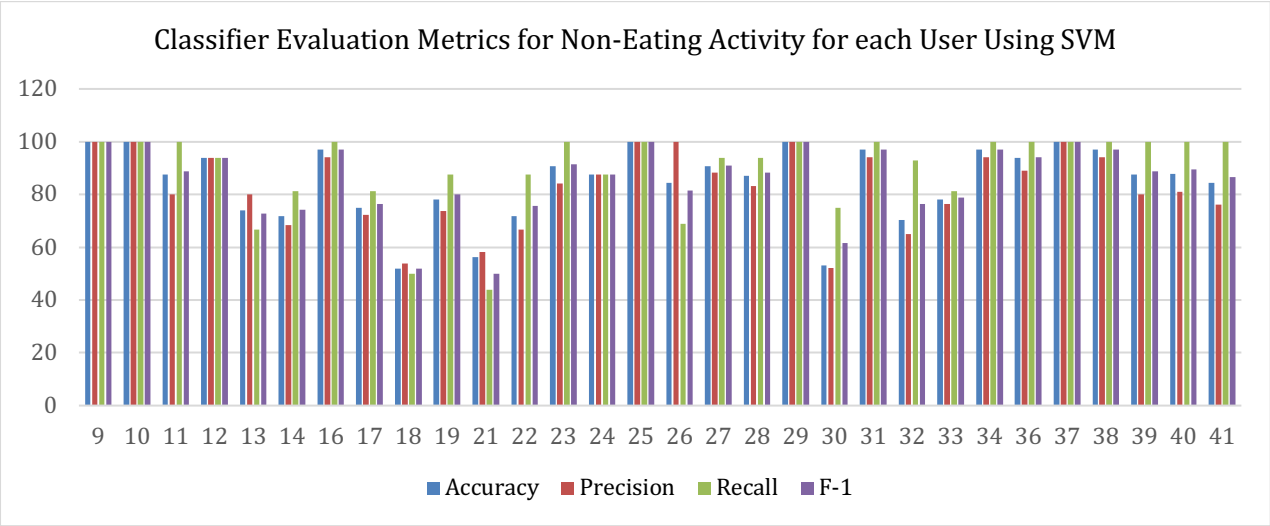


Table showing the statistics for evaluation metrics for each user for **Non-Eating Activity**

User	Accuracy	Precision	Recall	F-1
9	100	100	100	100
10	100	100	100	100
11	87.5	80	100	88.8888889
12	93.8	93.8	93.8	93.8
13	73.9	80	66.7	72.7471029
14	71.9	68.4	81.2	74.2524064
16	96.9	94.1	100	96.9603297
17	75	72.2	81.2	76.4359844
18	51.9	53.8	50	51.8304432
19	78.1	73.7	87.5	80.0093052
21	56.2	58.3	43.8	50.0203722
22	71.9	66.7	87.5	75.6971466
23	90.6	84.2	100	91.422367
24	87.5	87.5	87.5	87.5
25	100	100	100	100
26	84.4	100	68.8	81.5165877
27	90.6	88.2	93.8	90.9138462
28	87.1	83.3	93.8	88.2387352
29	100	100	100	100

30	53.1	52.2	75	61.5566038
31	96.9	94.1	100	96.9603297
32	70.4	65	92.9	76.4851172
33	78.1	76.5	81.2	78.779962
34	96.9	94.1	100	96.9603297
36	93.8	88.9	100	94.1238751
37	100	100	100	100
38	96.9	94.1	100	96.9603297
39	87.5	80	100	88.8888889
40	87.9	81	100	89.5027624
41	84.4	76.2	100	86.492622



Neural Net

User Dependent Analysis

Table showing the statistics for evaluation metrics for each user for **Eating Activity**

User	Accuracy	Precision	Recall	F-1
9	100	100	100	100
10	96.6	93.3	100	96.5338852
11	93.8	88.9	100	94.1238751
12	100	100	100	100
13	73.9	66.7	90.9	76.9420051
14	78.1	84.6	68.8	75.8863103
16	100	100	100	100
17	78.1	80	75	77.4193548
18	59.3	58.3	53.8	55.9596789
19	96.9	94.1	100	96.9603297
21	56.2	55	68.8	61.1308562
22	68.8	71.4	62.5	66.6542196

23	100	100	100	100
24	68.8	63.6	87.5	73.6598279
25	96.6	94	100	96.9072165
26	93.8	93.8	93.8	93.8
27	87.5	87.5	87.5	87.5
28	93.5	93.5	93.5	93.5
29	96.9	94.1	100	96.9603297
30	71.9	64	100	78.0487805
31	100	100	100	100
32	96.3	92.9	100	96.3193364
33	68.8	66.7	75	70.606916
34	100	100	100	100
36	100	100	100	100
37	100	100	100	100
38	93.8	93.8	93.8	93.8
39	96.9	94.1	100	96.9603297
40	100	100	100	100
41	90.6	93.3	87.5	90.306969

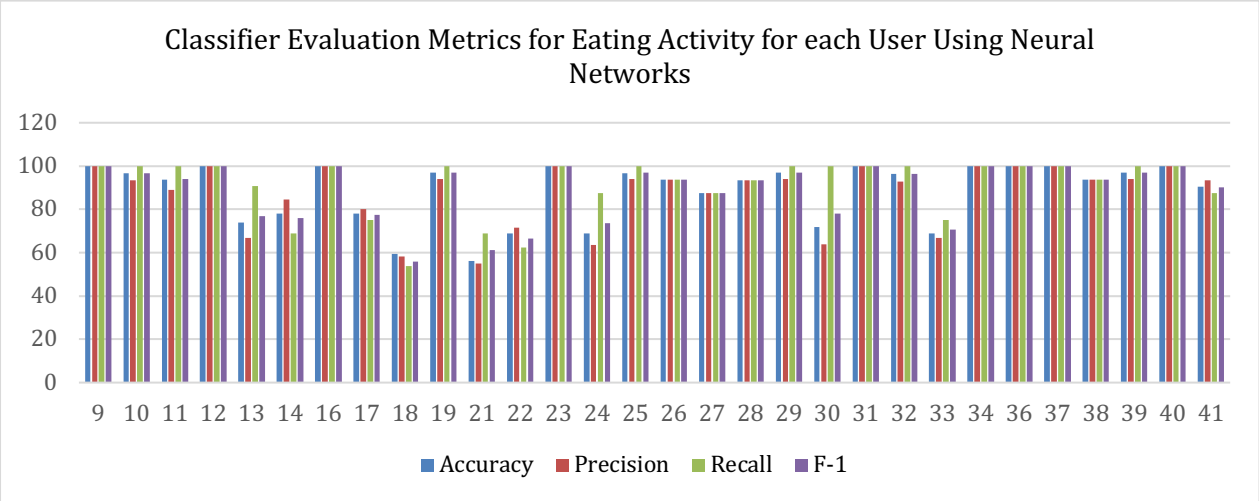
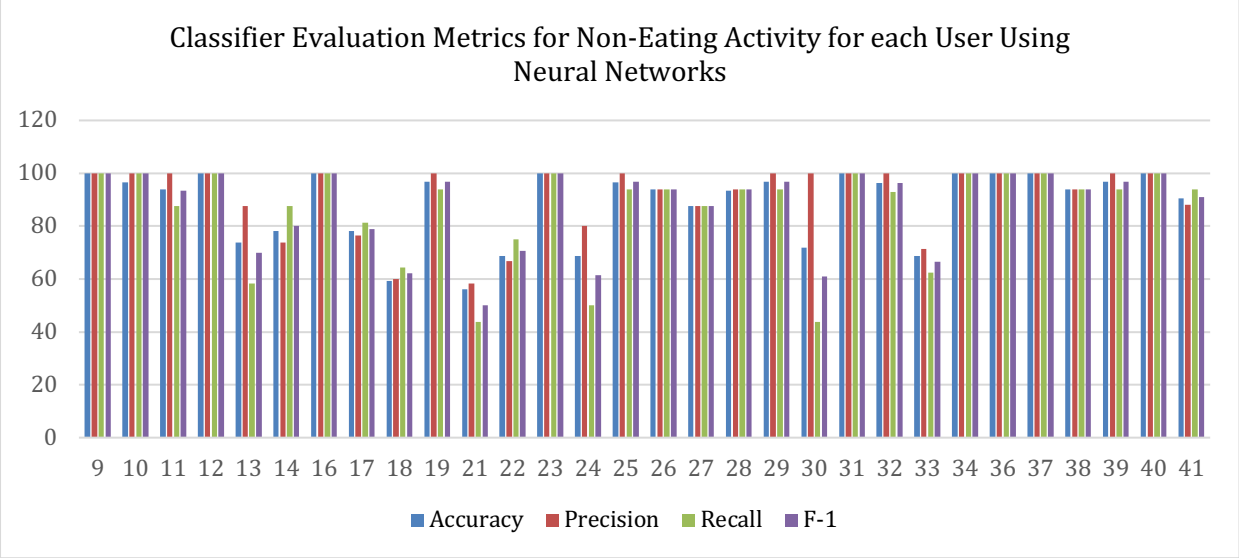


Table showing the statistics for evaluation metrics for each user for **Non-Eating Activity**

User	Accuracy	Precision	Recall	F-1
9	100	100	100	100
10	96.6	100	100	100
11	93.8	100	87.5	93.3333333
12	100	100	100	100
13	73.9	87.5	58.3	69.9759945
14	78.1	73.7	87.5	80.0093052
16	100	100	100	100
17	78.1	76.5	81.2	78.779962
18	59.3	60	64.3	62.0756235
19	96.9	100	93.8	96.8008256

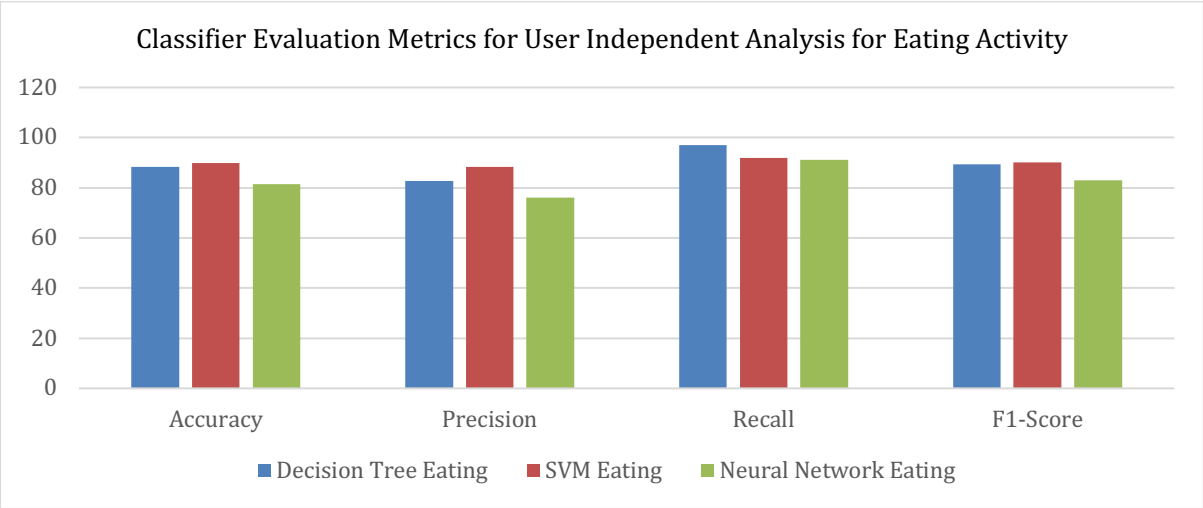
21	56.2	58.3	43.8	50.0203722
22	68.8	66.7	75	70.606916
23	100	100	100	100
24	68.8	80	50	61.5384615
25	96.6	100	93.8	96.8008256
26	93.8	93.8	93.8	93.8
27	87.5	87.5	87.5	87.5
28	93.5	93.8	93.8	93.8
29	96.9	100	93.8	96.8008256
30	71.9	100	43.8	60.9179416
31	100	100	100	100
32	96.3	100	92.9	96.3193364
33	68.8	71.4	62.5	66.6542196
34	100	100	100	100
36	100	100	100	100
37	100	100	100	100
38	93.8	93.8	93.8	93.8
39	96.9	100	93.81	96.8061504
40	100	100	100	100
41	90.6	88.2	93.8	90.9138462

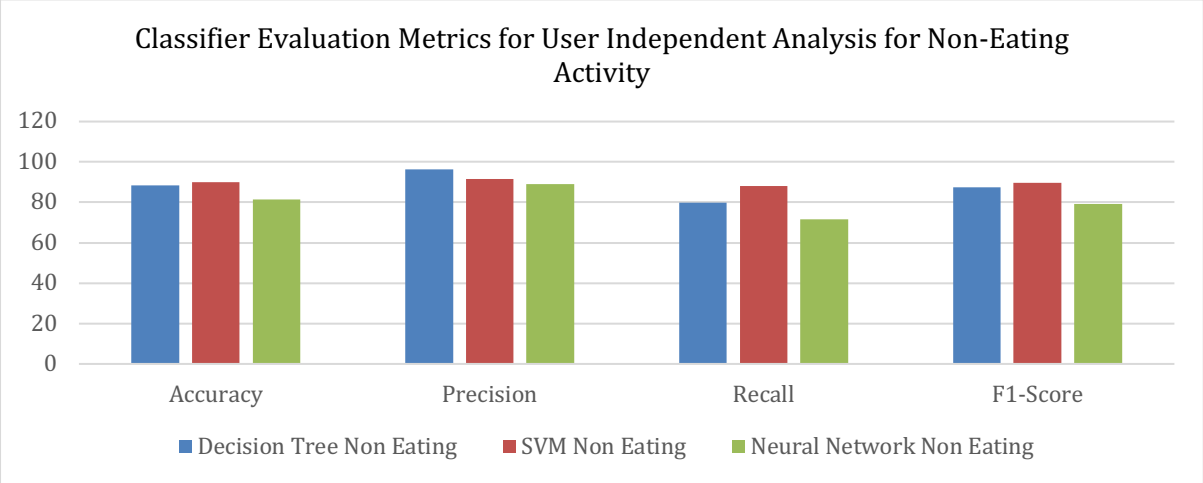


Classifier Evaluation Metrics – Results for User Independent Analysis

Table showing the statistics for evaluation metrics for all 3 classifiers (Decision Tree, SVM, Neural Net) for both eating and non-eating activities

Classifier	Activity	Accuracy	Precision	Recall	F1-Score
Decision Tree	Eating	88.4	82.7	97	89.28102
Decision Tree	Non-Eating	88.4	96.4	79.8	87.31805
SVM	Eating	89.9	88.4	91.8	90.06792
SVM	Non-Eating	89.9	91.5	88	89.71588
Neural Network	Eating	81.4	76.2	91.2	83.02796
Neural Network	Non-Eating	81.4	89	71.6	79.35741





References

1. https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1
2. https://www.mathworks.com/help/stats/fitctree.html
3. https://www.mathworks.com/help/stats/fitcsvm.html
4. https://www.mathworks.com/help/stats/compactclassificationdiscriminant.predict.html
5. https://www.mathworks.com/help/deeplearning/ref/train.html
6. https://www.mathworks.com/help/deeplearning/ref/plotconfusion.html
7. https://www.mathworks.com/help/deeplearning/ref/patternnet.html
8. https://en.wikipedia.org/wiki/Precision_and_recall
9. https://en.wikipedia.org/wiki/F1_score