

# Homework 2

Harshith Bondada  
M10724854

**1. Take Data2 and split it into a Training Set of 210 randomly selected data points. Use the remaining 100 data points as the Testing Set. Train a linear support vector machine (kernel\_function = linear) to build a classifier model. Use this model to predict the classes for the data in the Testing Set. Report the following:**

## **Solution:**

The below code is for building a classifier. I have taken 210 observations as testing data and 100 observations as training data.

I have converted them into a table so that svm train can work on the tables.

I have used svm train function to generate a classifier with the training data using the kernel function 'linear'.

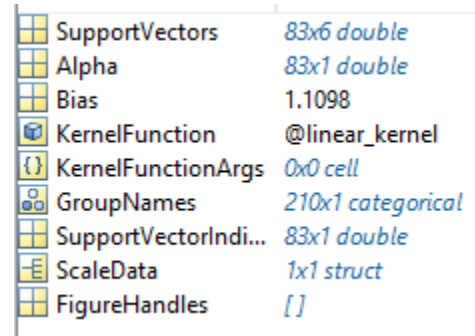
I have predicted them against the test class using the svm classify function and initialized the values in the 'groups'.

```
load Data2.mat
k = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training = Data2 (k(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing = Data2 (k(211: end), :); % Dividing the dataset2 into 100 test
rows

Data2_training_mat = table2array(Data2_training(:,1:6));
training_class_mat = table2array(Data2_training(:,7));
Data2_testing_mat = table2array(Data2_testing(:,1:6));
testing_class_mat = table2array(Data2_testing(:,7));

svmStruct = svmtrain(Data2_training_mat ,training_class_mat,
'kernel_function', 'linear');
groups = svmclassify(svmStruct, Data2_testing_mat);
```

The below is the screenshot of the classifier which is generated in the matlab. We can see that there are 83 support vectors.



A screenshot of the MATLAB variable inspector window, which displays the properties of a trained classifier. The window has a vertical toolbar on the left with icons for variable types: matrix, cell, struct, function handle, and class. The main area lists the following variables and their types:

SupportVectors	83x6 double
Alpha	83x1 double
Bias	1.1098
KernelFunction	@linear_kernel
KernelFunctionArgs	0x0 cell
GroupNames	210x1 categorical
SupportVectorIndi...	83x1 double
ScaleData	1x1 struct
FigureHandles	[]

1a . List all the support vectors of the data set. Examine the number of support vectors for each class and their attribute values and comment on what can be inferred from these values.

Solution:

The classifier has generated 83 support vectors across 6 columns.

SupportVectors *83x6 double*

	1	2	3	4	5	6
1	-1.2552	-0.3933	-1.0907	-1.3241	0.3841	-0.6295
2	0.8716	0.4622	1.6762	0.7829	0.4398	-0.4111
3	-0.8646	0.1421	-0.9430	-1.2103	-0.0739	-0.5882
4	-0.3434	-0.7653	-0.1855	0.1127	0.3754	-0.4643
5	-0.8183	-0.6317	-1.0302	-0.5923	-1.3715	-0.5092
6	-1.3436	-0.0874	-1.4634	-1.6584	0.5177	-0.4854
7	-1.0808	0.0555	-1.6748	-1.4248	0.0389	-0.6141
8	0.2690	0.6860	-0.3039	-0.1508	1.3403	-0.2668
9	0.5367	-0.7061	1.3252	1.1975	0.1018	0.1284
10	0.3827	-0.2714	0.3471	0.6862	0.4182	-0.0341
11	-0.9584	-0.3194	-0.7269	-0.9973	1.1978	-0.5076
12	-0.9228	0.4617	-1.3151	-1.5157	-0.4860	-0.5998
13	0.1724	0.3037	0.1635	0.0015	-0.3970	-0.6793
14	-1.6648	-0.9594	-1.4483	-1.4401	-0.2873	-0.5576
15	0.6679	0.0037	0.0275	0.8530	0.0986	0.1149
16	0.1994	0.9047	-0.0279	-0.3978	-0.8385	-0.5885
17	-0.1250	-1.7742	9.1493e-04	1.1212	-1.1362	0.0166
18	-0.2709	-0.1059	0.5747	-0.2706	-0.2736	-0.7206
19	0.4825	0.7359	-0.3819	0.0868	-1.1487	-0.3921
20	-0.7414	-0.5299	-0.6647	-0.5672	-0.0224	-0.1381
21	-1.4372	-0.5689	-1.9112	-1.4305	-0.7837	-0.7481
22	1.6489	0.5330	2.0810	1.7276	-1.2472	-0.5904
23	-0.6231	-0.0617	-1.2504	-0.7538	0.2600	-0.6177
24	-0.3791	0.9372	-0.8601	-1.1625	0.2609	-0.6260
25	-0.8829	-0.8720	-0.5271	-0.5014	0.0520	-0.6589
26	-1.2176	-1.0415	-0.7712	-0.8079	-3.5485e-04	-0.6180
27	-1.6767	-1.4311	-1.0075	-1.1147	0.8055	-0.5763
28	-0.3912	-0.4442	-0.6647	-0.1804	0.0266	-0.5410
29	-0.6814	0.0674	0.0275	-0.9217	1.5355	-0.4108
30	-0.0684	1.4760	-0.2621	-1.1535	0.1085	-0.6284
31	-1.3003	-1.2966	-1.4256	-0.7295	2.8817	0.1512
32	-0.2382	-0.0552	-0.5049	-0.2653	0.6819	-0.0814
33	0.5899	0.6597	-0.4146	0.2793	0.1470	-0.0022
34	-1.0000	-0.7427	-1.2005	-0.7448	0.4063	-0.6219
35	-1.3844	-0.3495	-1.6635	-1.5213	0.5942	-0.6598

The support vectors and their attribute values are represented through these screen shots.

36	-0.6749	0.2636	-0.5974	-1.0550	0.1078	-0.4694
37	-1.0384	-0.0813	-0.5049	-1.2716	0.2022	-0.4730
38	0.0421	0.2346	0.0750	-0.1155	0.0621	-0.5473
39	-0.3944	0.1896	-0.9960	-0.6423	0.2770	-0.5348
40	-1.4013	0.5451	-1.1972	-2.1889	-0.1617	-0.7427
41	-0.9660	-0.3843	-0.4682	-0.9600	0.5169	-0.3417
42	-0.7655	-0.8841	-0.8479	-0.3424	0.1982	-0.6246
43	-0.3340	0.3742	-0.4943	-0.6981	0.5320	-0.6051
44	1.2554	1.2640	0.3604	0.6956	-0.5122	-0.5167
45	1.0047	0.4870	0.4535	0.9356	0.0509	2.2264e-04
46	-1.0664	-2.4083	0.8742	0.3729	-0.4491	-8.5805e-04
47	0.3115	0.9047	-0.2121	-0.2542	0.2445	-0.6836
48	-0.4184	-0.1495	-0.7624	-0.4280	0.1975	-0.5189
49	-0.3895	-0.8102	-0.4463	0.0861	0.4733	-0.0556
50	-1.9679	-0.6639	-1.9959	-2.0418	0.5313	-0.9090
51	0.1966	-0.2872	-0.4036	0.4593	-0.1753	-0.5198
52	-0.2667	-0.4274	0.5939	-0.0330	-0.1148	0.0922
53	-0.3521	0.4172	-1.1780	-0.7523	0.0374	-0.6998
54	-0.9910	-0.9955	-1.2469	-0.5507	-0.3634	-0.5246
55	-1.5310	-1.2367	-0.7903	-1.0684	0.4407	-0.6690
56	0.1265	0.5243	-0.6323	-0.2165	-1.3788	-0.6703
57	0.3452	0.7600	-0.1174	-0.1065	-0.3129	-0.7128
58	0.1264	1.0063	0.0065	-0.5647	-0.2388	-0.4836
59	-0.2075	-0.0866	-0.4601	-0.2033	-0.2893	-0.5425
60	-1.0383	-0.2995	-1.3930	-1.1139	0.7979	-0.6458
61	0.2390	1.0331	-0.0285	-0.4399	-0.0895	-0.4939
62	-0.8666	-0.4311	-1.1281	-0.7989	0.0113	-0.7852
63	-0.5265	-0.3860	-0.8777	-0.3958	0.6404	-0.3262
64	0.7982	-0.3465	0.5969	1.2728	-0.1883	0.1407
65	-1.1670	-0.3452	-1.4035	-1.2459	0.9003	-0.6100
66	-0.6756	-1.1849	0.2139	-0.0099	1.3884	-0.1824
67	-0.9240	-0.7975	0.0275	-0.6079	1.2159	0.0420
68	-1.0830	-0.5133	-1.1374	-1.0168	-0.0891	-0.6943
69	-0.2815	1.1585	-0.2019	-1.1973	0.3951	-0.5959
70	-0.7919	-0.2012	-0.7179	-0.8693	-0.1132	-0.6238

71	-0.9263	0.4471	-1.0654	-1.5096	-0.2912	-0.6710
72	-1.6791	0.0370	-1.9160	-2.1780	0.1607	-0.6520
73	-0.7080	0.4916	-0.8148	-1.2621	-0.0316	-0.5069
74	-0.5520	-0.3162	-0.8270	-0.4789	-0.1454	-0.4966
75	-0.6300	-0.4340	-1.0729	-0.4937	-0.6606	-0.8540
76	-1.2348	-0.7342	-1.4094	-1.0518	-0.2461	-0.5534
77	-0.0216	-1.2174	0.4436	0.8515	-0.6329	0.0702
78	-1.6304	-1.0437	-0.8245	-1.3351	1.0397	-0.5086
79	-0.9009	0.0096	-1.2621	-1.1611	0.0113	-0.5290
80	-0.8721	-0.7498	-0.3452	-0.5758	3.2577	-0.1713
81	0.8234	-0.0682	0.5226	1.1042	0.0763	0.1019
82	-0.7324	-0.3760	-0.8244	-0.6668	-0.0270	-0.7672
83	-0.6296	-0.7754	-1.2335	-0.2467	-0.6951	-0.4720

**1b. Run the training and testing five times, each time selecting a different randomly selected set of training instances. Create the confusion matrix for this classifier using average number of true positives, false positives etc.**

Solution:

I have generated training and testing data 5 times and build 5 svm classifier models and calculated their True positive/negative and False positive/negative against the predicted values.

Below is the entire code for building a classifier and then predicting against the test values and calculating their TP's, FP's, TN's, FN's.

```
%Question 1b. Building 5 training sets
%Generating 1st SVM
k1 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training1 = Data2 (k1(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing1 = Data2 (k1(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat1 = table2array(Data2_training1(:,1:6));
training_class_mat1 = table2array(Data2_training1(:,7));
Data2_testing_mat1 = table2array(Data2_testing1(:,1:6));
testing_class_mat1 = table2array(Data2_testing1(:,7));

svmStruct1 = svmtrain(Data2_training_mat1 ,training_class_mat1,
'kernel_function', 'linear');
groups1 = svmclassify(svmStruct1, Data2_testing_mat1);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP1 = sum(groups1 == "Abnormal" & testing_class_mat1=="Normal");
FP1 = sum(groups1 == "Normal" & testing_class_mat1=="Abnormal");
TN1 = sum(groups1 == "Abnormal" & testing_class_mat1=="Abnormal");
```

```

FN1 = sum(groups1 == "Abnormal" & testing_class_mat1=="Normal");

%Generating 2nd SVM
k2 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training2 = Data2 (k2(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing2 = Data2 (k2(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat2 = table2array(Data2_training2(:,1:6));
training_class_mat2 = table2array(Data2_training2(:,7));
Data2_testing_mat2 = table2array(Data2_testing2(:,1:6));
testing_class_mat2 = table2array(Data2_testing2(:,7));

svmStruct2 = svmtrain(Data2_training_mat2 ,training_class_mat2,
'kernel_function', 'linear');
groups2 = svmclassify(svmStruct2, Data2_testing_mat2);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP2 = sum(groups2 == "Abnormal" & testing_class_mat2=="Normal");
FP2 = sum(groups2 == "Normal" & testing_class_mat2=="Abnormal");
TN2 = sum(groups2 == "Abnormal" & testing_class_mat2=="Abnormal");
FN2 = sum(groups2 == "Abnormal" & testing_class_mat2=="Normal");

%Generating 3rd SVM
k3 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training3 = Data2 (k3(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing3 = Data2(k3(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat3 = table2array(Data2_training3(:,1:6));
training_class_mat3 = table2array(Data2_training3(:,7));
Data2_testing_mat3 = table2array(Data2_testing3(:,1:6));
testing_class_mat3 = table2array(Data2_testing3(:,7));

svmStruct3 = svmtrain(Data2_training_mat3 ,training_class_mat3,
'kernel_function', 'linear');
groups3 = svmclassify(svmStruct3, Data2_testing_mat3);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP3 = sum(groups3 == "Abnormal" & testing_class_mat3=="Normal");
FP3 = sum(groups3 == "Normal" & testing_class_mat3=="Abnormal");
TN3 = sum(groups3 == "Abnormal" & testing_class_mat3=="Abnormal");
FN3 = sum(groups3 == "Abnormal" & testing_class_mat3=="Normal");

```

```

%Generating 4th SVM
k4 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training4 = Data2 (k4(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing4 = Data2(k4(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat4 = table2array(Data2_training4(:,1:6));
training_class_mat4 = table2array(Data2_training4(:,7));
Data2_testing_mat4 = table2array(Data2_testing4(:,1:6));
testing_class_mat4 = table2array(Data2_testing4(:,7));

svmStruct4 = svmtrain(Data2_training_mat4 ,training_class_mat4,
'kernel_function', 'linear');
groups4 = svmclassify(svmStruct4, Data2_testing_mat4);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP4 = sum(groups4 == "Abnormal" & testing_class_mat4=="Normal");
FP4 = sum(groups4 == "Normal" & testing_class_mat4=="Abnormal");
TN4 = sum(groups4 == "Abnormal" & testing_class_mat4=="Abnormal");
FN4 = sum(groups4 == "Abnormal" & testing_class_mat4=="Normal");

%Generating 5th SVM
k5 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training5 = Data2 (k5(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing5 = Data2(k5(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat5 = table2array(Data2_training5(:,1:6));
training_class_mat5 = table2array(Data2_training5(:,7));
Data2_testing_mat5 = table2array(Data2_testing5(:,1:6));
testing_class_mat5 = table2array(Data2_testing5(:,7));

svmStruct5 = svmtrain(Data2_training_mat5 ,training_class_mat5,
'kernel_function', 'linear');
groups5 = svmclassify(svmStruct5, Data2_testing_mat5);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP5 = sum(groups5 == "Abnormal" & testing_class_mat5=="Normal");
FP5 = sum(groups5 == "Normal" & testing_class_mat5=="Abnormal");
TN5 = sum(groups5 == "Abnormal" & testing_class_mat5=="Abnormal");
FN5 = sum(groups5 == "Abnormal" & testing_class_mat5=="Normal");

```

Below is the code for calculating the average values of the TP's, TN's, FP's, FN's.

```
%Computing average of the True/False positive and True/False negative
Avg_TP= (TP1+TP2+TP3+TP4+TP5) /5;
Avg_FP= (FP1+FP2+FP3+FP4+FP5) /5;
Avg_TN= (TN1+TN2+TN3+TN4+TN5) /5;
Avg_FN= (FN1+FN2+FN3+FN4+FN5) /5;
```

Below is the code for building a confusion matrix.

```
%Creating confusion matrix based on the average performance values
C= [Avg_TN Avg_FP; Avg_FN Avg_TP];
```

	1	2
1	53.2000	14
2	2.8000	2.8000

The above fig represents the confusion matrix.

### 1c. Compute the accuracy, precision and recall values.

Solution:

I have calculated the accuracy, precision, and recall using the formula based on the confusion matrix.

```
accuracy=(C(1,1)+C(2,2))/(C(1,1)+C(1,2)+C(2,1)+C(2,2));
precision=C(2,2)/(C(2,2)+C(1,2));
recall=(C(2,2)/(C(2,2)+C(2,1)));
```

```
accuracy =

    0.7692

>> precision

precision =

    0.1667

>> recall

recall =

    0.5000
```



## 2. Repeat Q#1 above with the only difference that this time train a non-linear SVM using rbf (radial basis function) for kernel function.

Solution:

The below code is for building a classifier. I have taken 210 observations as testing data and 100 observations as training data.

I have converted them into a table so that svm train can work on the tables.

I have used svm train function to generate a classifier with the training data using the kernel function 'rbf'.

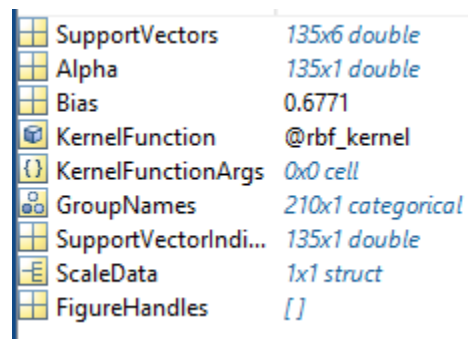
I have predicted them against the test class using the svm classify function and initialized the values in the 'groups'.

```
%-----Question 2-----
load Data2.mat
k = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training = Data2 (k(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing = Data2 (k(211: end), :); % Dividing the dataset2 into 100 test
rows

Data2_training_mat = table2array(Data2_training(:,1:6));
training_class_mat = table2array(Data2_training(:,7));
Data2_testing_mat = table2array(Data2_testing(:,1:6));
testing_class_mat = table2array(Data2_testing(:,7));

svmStruct = svmtrain(Data2_training_mat ,training_class_mat,
'kernel_function', 'rbf');
groups = svmclassify(svmStruct, Data2_testing_mat);
```

The below is the screenshot of the classifier which is generated in the matlab. We can see that there are support vectors.



SupportVectors	135x6 double
Alpha	135x1 double
Bias	0.6771
KernelFunction	@rbf_kernel
KernelFunctionArgs	0x0 cell
GroupNames	210x1 categorical
SupportVectorIndi...	135x1 double
ScaleData	1x1 struct
FigureHandles	[]

The below screenshots are the 135 support vectors with their attribute value across the 6 columns.

	1	2	3	4	5	6
1	0.8417	1.3849	1.9430	0.0117	-0.7891	0.6559
2	-6.9406e-04	0.1883	-0.0466	-0.1404	0.0896	-0.5634
3	-1.2650	-0.8738	-1.2321	-0.9125	1.6038	-0.7416
4	-0.7668	-1.2330	0.0947	-0.0318	1.4548	-0.1978
5	-1.0509	-1.6447	-0.4142	-0.0771	-0.6557	0.3604
6	-1.0767	-0.4315	-0.5996	-1.0081	0.5577	-0.3574
7	-0.8196	-1.3962	0.2489	0.0239	0.5835	0.1705
8	0.1099	0.2233	0.7450	-0.0300	1.0021	0.2326
9	4.1391	-0.9489	-0.2910	5.8077	-0.7253	9.4958
10	-0.6348	-0.3632	-0.9649	-0.5137	-0.1240	-0.5126
11	0.0895	0.4783	-0.7666	-0.2442	-1.3935	-0.6867
12	-1.0446	-0.5319	-0.9569	-0.8941	0.4920	-0.5488
13	-0.1651	1.9941	3.9019	-1.6813	1.3452	2.1718
14	0.7888	-0.2105	0.2637	1.1288	-0.8937	-0.6659
15	-0.6450	0.5250	-0.3660	-1.1844	-0.0041	-0.7419
16	0.4280	-0.2752	0.5020	0.7318	0.4412	0.2798
17	-0.2998	-0.1019	-0.6370	-0.2942	0.7275	-0.0965
18	-0.7729	0.0209	-0.0950	-0.9686	1.6061	-0.4266
19	0.3613	1.5601	2.3052	-0.7105	2.0842	1.4664
20	0.9508	-0.1083	-0.2156	1.2528	-0.5030	0.2869
21	-1.1540	-0.1281	-0.6370	-1.3282	0.2338	-0.4890
22	-0.4502	0.8917	-0.9986	-1.2160	0.2942	-0.6423
23	0.0894	0.9609	-0.1164	-0.6019	-0.2202	-0.4996
24	0.7983	0.1963	1.5709	0.8390	-0.6018	0.1293
25	-0.5536	0.1211	-0.9691	-0.7725	-0.0680	-0.6393
26	0.2869	0.8592	-0.3389	-0.2829	0.2773	-0.6999
27	0.1642	-0.3342	-0.5339	0.4502	-0.1548	-0.5358
28	1.2944	1.2189	0.2438	0.6930	-0.5016	-0.5327
29	-1.3854	-0.4404	-1.2332	-1.3821	0.4211	-0.6458
30	0.5531	0.3928	0.2942	0.3911	-1.0723	-0.0043
31	-0.0484	-0.2075	-0.5720	0.0941	-0.3279	0.0901
32	0.1459	-0.1765	0.5079	0.3108	1.8440	-0.5274
33	0.1975	-0.2660	0.2757	0.4406	-0.4296	-0.1600
34	-0.8022	-0.0362	-0.3118	-0.9624	0.8525	-0.7027

35	-1.1123	0.1869	-1.0164	-1.5102	0.5461	-0.7515
36	-0.7660	0.2173	-0.7311	-1.1057	0.1367	-0.4853
37	-0.4922	-0.1964	-0.8990	-0.4614	0.2290	-0.5349
38	1.1782	1.9199	1.7999	0.0302	1.7497	1.4074
39	0.5728	1.9878	1.6685	-0.7666	0.5697	0.9673
40	-0.2671	-0.1333	-0.5914	-0.2306	-0.2721	-0.5586
41	2.0567	2.8992	0.2844	0.3879	-1.5409	1.1985
42	-1.4249	-0.9593	-1.4914	-1.0463	0.4688	-0.5860
43	1.6648	0.8338	0.8276	1.4351	-0.4444	-0.5331
44	0.3629	-0.3183	0.2303	0.6834	0.4561	-0.0491
45	-0.1071	0.0069	0.2956	-0.1372	0.1365	-0.1426
46	0.8137	-0.2914	-0.0012	1.2195	-0.8768	-0.6417
47	0.6380	0.1145	-0.1492	0.7019	-0.2419	-0.6559
48	-1.5073	-1.2912	-0.6398	-0.9020	-2.4630	-0.6643
49	1.3567	2.3682	0.4470	-0.0818	-0.1827	-0.0341
50	-0.7177	-0.8230	-1.3786	-0.2751	-0.6898	-0.4880
51	1.0728	0.6188	-0.7022	0.8644	-1.3973	0.2120
52	-1.1993	0.0089	-1.8278	-1.4856	0.0657	-0.6303
53	-1.0086	-0.7658	-0.4944	-0.6763	0.9336	0.2280
54	-1.5233	-0.3966	-1.8163	-1.5847	0.6373	-0.6761
55	-1.3272	-0.1616	-0.9257	-1.5170	1.0705	-0.8013
56	-1.0343	0.4011	-1.2075	-1.5728	-0.2741	-0.6874
57	-0.5666	-0.0603	1.3194	-0.6541	1.4447	0.6553
58	-0.6950	-0.4419	-1.0458	-0.5297	0.1228	-0.6054
59	-1.2727	0.0168	-1.0706	-1.5820	0.2645	-0.7179
60	-1.7282	-1.8295	-1.8798	-0.7756	0.2157	-0.4531
61	-1.0319	-0.8451	-0.0950	-0.6463	1.2772	0.0272
62	-2.1461	-0.7113	-2.1547	-2.1195	0.5725	-0.9259
63	-0.3347	-0.1526	0.4619	-0.2997	-0.2560	-0.7370
64	0.2415	0.6402	-0.4324	-0.1766	1.4053	-0.2823
65	-0.6906	-2.0963	-0.6370	0.7019	1.0616	-0.8821
66	-1.1174	-0.4030	-0.7266	-1.0794	0.8182	-0.6569
67	-1.3453	-1.0894	-0.9080	-0.8518	0.0253	-0.6342
68	0.3004	0.3114	-1.1938	0.1397	0.8467	-0.6436

69	-0.3798	0.2652	-1.0706	-0.6650	-0.1186	-0.5940
70	-1.2344	-1.2123	-0.5900	-0.6240	-1.0470	-0.0080
71	-0.8370	-0.5772	-0.7996	-0.6045	0.0026	-0.1533
72	-1.8227	-1.0072	-1.5972	-1.5013	-0.2701	-0.5737
73	2.1280	0.1665	1.9761	2.5009	0.2368	0.8775
74	1.1250	1.6273	1.6991	0.1815	0.6015	1.7580
75	1.0214	-0.3802	1.4061	1.5413	-0.9446	-0.3792
76	0.5699	0.2370	0.4936	0.5271	-0.0963	0.1102
77	-0.9765	-0.7974	-0.4744	-0.6133	3.3788	-0.1866
78	-0.4128	-0.5766	-0.6912	-0.0818	0.0123	0.3013
79	-0.6503	-0.8823	0.1381	-0.1481	-2.8515	-0.1203
80	0.4695	0.6902	-0.5117	0.0675	-1.1567	-0.4079
81	1.6945	3.1366	0.9892	-0.2347	1.2715	2.1971
82	-1.2354	-0.0494	-1.0994	-1.4870	-0.0815	-0.9132
83	-0.2443	1.5975	-0.1541	-1.4851	1.7418	2.9362
84	-0.7107	-0.1084	-1.3958	-0.7961	0.2933	-0.6340
85	-0.3661	0.2304	-0.5286	-0.6223	0.3832	0.1596
86	-1.5413	0.4991	-1.3416	-2.2707	-0.1408	-0.7592
87	0.0163	-0.4251	0.5554	0.3352	0.3188	-0.6490
88	0.7809	2.3955	-1.4121	-0.8122	-0.7061	-0.5589
89	-0.3923	-2.1760	0.1218	1.1288	-0.5604	0.0920
90	1.3963	1.2661	0.6356	0.7836	-0.7016	-0.0698
91	0.2095	0.9878	-0.1521	-0.4736	-0.0664	-0.5099
92	-0.6076	-0.4331	-1.0164	-0.4283	0.6848	-0.3419
93	0.8333	-0.1149	0.4089	1.1128	0.1042	0.0872
94	-1.7859	-1.0916	-0.9623	-1.3934	1.0959	-0.5246
95	1.7556	1.0554	0.8705	1.3830	-1.2291	0.7497
96	-0.4770	-0.4402	-0.6596	-0.2620	-0.2641	-0.6140
97	-0.7181	-0.4812	-1.2151	-0.5289	-0.6543	-0.8708
98	0.1384	0.2574	0.0433	-0.0201	-0.3830	-0.6956
99	-0.9811	-0.7117	-1.3396	-0.6825	-0.0154	-0.9401
100	3.4305	2.0820	-0.1580	2.6878	-2.7014	1.1198
101	0.4988	-0.4069	0.2302	0.9167	0.1077	-0.2449

102	0.4757	0.1097	1.2040	0.5053	-1.0297	-0.6913
103	0.0922	0.6661	0.0134	-0.3799	-0.8191	-0.2968
104	-1.1839	-2.4580	0.7668	0.3615	-0.4366	-0.0158
105	-0.8013	0.4456	-0.9524	-1.3184	-0.0069	-0.5229
106	0.7830	1.5608	0.7149	-0.1910	2.1483	2.3583
107	0.2461	0.5374	0.4786	-0.0947	0.4929	-0.7798
108	0.5363	-0.4379	0.4036	0.9859	-1.1220	-0.0596
109	0.7325	-0.8171	0.5012	1.5088	-1.3895	-0.0246
110	0.8847	0.4162	1.5831	0.7827	0.4783	-0.4269
111	-0.6592	-1.1449	-1.0164	0.0354	-0.0659	-0.6857
112	-0.1837	-0.3923	-0.6659	0.0641	-0.1454	0.0583
113	-1.0894	-0.1313	-0.3660	-1.2461	-0.6081	-0.4619
114	0.4413	0.4446	-0.1984	0.2147	-0.8517	-0.7663
115	-0.9706	-0.4783	-1.2713	-0.8425	0.0373	-0.8018
116	0.5255	2.2205	0.7786	-0.9974	2.3009	2.8542
117	-1.1296	-2.3864	0.2302	0.3755	0.3060	-0.7622
118	-0.9159	-0.9498	-1.3396	-0.4257	-0.1479	-0.6308
119	-0.4021	0.3280	-0.6262	-0.7389	0.5732	-0.6214
120	1.1839	0.3475	1.3558	1.2025	-2.0309	0.5147
121	-1.3875	-0.4867	-0.7454	-1.3504	0.5511	-0.6148
122	-0.8149	-0.2905	-0.9858	-0.7896	0.4876	-0.4873
123	0.3184	-0.3288	0.9936	0.6363	-2.5953	0.3330
124	1.9775	-0.2451	0.7563	2.6204	-0.1917	0.6143
125	1.1852	0.7045	1.3081	0.9396	2.5465	0.9051
126	-0.8909	-0.2481	-0.8538	-0.9148	-0.0909	-0.6401
127	-0.0064	0.4927	-0.4110	-0.3730	0.6072	-0.7463
128	-1.0306	0.4157	-1.4616	-1.5790	-0.4746	-0.6160
129	0.1441	0.2179	-0.7358	0.0162	-0.3269	-0.9494
130	-1.0687	-0.3665	-0.8629	-1.0463	1.2586	-0.5236
131	-1.3636	-0.7818	-1.5576	-1.1023	-0.2277	-0.5695
132	1.1146	0.0972	0.5181	1.3025	-0.1994	0.2565
133	1.8473	1.7736	1.2826	0.9638	-0.1300	0.7311
134	0.6542	-0.1402	0.3315	0.9106	-3.5123	-0.3870
135	-0.9685	0.0957	-1.0829	-1.2652	-0.0504	-0.6044

**2a.Run the training and testing five times, each time selecting a different randomly selected set of training instances. Create the confusion matrix for this classifier using average number of true positives, false positives etc.**

Solution:

I have generated training and testing data 5 times and build 5 svm classifier models and calculated their True positive/negative and False positive/negative against the predicted values.

Below is the entire code for building a classifier and then predicting against the test values and calculating their TP's, FP's, TN's, FN's.

```
%Building 5 training sets
%Generating 1st SVM
k1 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training1 = Data2 (k1(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing1 = Data2 (k1(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat1 = table2array(Data2_training1(:,1:6));
training_class_mat1 = table2array(Data2_training1(:,7));
Data2_testing_mat1 = table2array(Data2_testing1(:,1:6));
testing_class_mat1 = table2array(Data2_testing1(:,7));

svmStruct1 = svmtrain(Data2_training_mat1 ,training_class_mat1,
'kernel_function', 'rbf');
groups1 = svmclassify(svmStruct1, Data2_testing_mat1);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP1 = sum(groups1 == "Abnormal" & testing_class_mat1=="Normal");
FP1 = sum(groups1 == "Normal" & testing_class_mat1=="Abnormal");
TN1 = sum(groups1 == "Abnormal" & testing_class_mat1=="Abnormal");
FN1 = sum(groups1 == "Abnormal" & testing_class_mat1=="Normal");

%Generating 2nd SVM
k2 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training2 = Data2 (k2(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing2 = Data2 (k2(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat2 = table2array(Data2_training2(:,1:6));
training_class_mat2 = table2array(Data2_training2(:,7));
Data2_testing_mat2 = table2array(Data2_testing2(:,1:6));
testing_class_mat2 = table2array(Data2_testing2(:,7));
```

```

svmStruct2 = svmtrain(Data2_training_mat2 ,training_class_mat2,
'kernel_function', 'rbf');
groups2 = svmclassify(svmStruct2, Data2_testing_mat2);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP2 = sum(groups2 == "Abnormal" & testing_class_mat2=="Normal");
FP2 = sum(groups2 == "Normal" & testing_class_mat2=="Abnormal");
TN2 = sum(groups2 == "Abnormal" & testing_class_mat2=="Abnormal");
FN2 = sum(groups2 == "Abnormal" & testing_class_mat2=="Normal");

%Generating 3rd SVM
k3 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training3 = Data2 (k3(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing3 = Data2(k3(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat3 = table2array(Data2_training3(:,1:6));
training_class_mat3 = table2array(Data2_training3(:,7));
Data2_testing_mat3 = table2array(Data2_testing3(:,1:6));
testing_class_mat3 = table2array(Data2_testing3(:,7));

svmStruct3 = svmtrain(Data2_training_mat3 ,training_class_mat3,
'kernel_function', 'rbf');
groups3 = svmclassify(svmStruct3, Data2_testing_mat3);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP3 = sum(groups3 == "Abnormal" & testing_class_mat3=="Normal");
FP3 = sum(groups3 == "Normal" & testing_class_mat3=="Abnormal");
TN3 = sum(groups3 == "Abnormal" & testing_class_mat3=="Abnormal");
FN3 = sum(groups3 == "Abnormal" & testing_class_mat3=="Normal");

%Generating 4th SVM
k4 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training4 = Data2 (k4(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing4 = Data2(k4(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat4 = table2array(Data2_training4(:,1:6));
training_class_mat4 = table2array(Data2_training4(:,7));
Data2_testing_mat4 = table2array(Data2_testing4(:,1:6));
testing_class_mat4 = table2array(Data2_testing4(:,7));

svmStruct4 = svmtrain(Data2_training_mat4 ,training_class_mat4,
'kernel_function', 'rbf');
groups4 = svmclassify(svmStruct4, Data2_testing_mat4);

```

```

TP4 = sum(groups4 == "Abnormal" & testing_class_mat4=="Normal");
FP4 = sum(groups4 == "Normal" & testing_class_mat4=="Abnormal");
TN4 = sum(groups4 == "Abnormal" & testing_class_mat4=="Abnormal");
FN4 = sum(groups4 == "Abnormal" & testing_class_mat4=="Normal");

%Generating 5th SVM
k5 = randperm(size(Data2,1)); % Creating a random permutation vector
Data2_training5 = Data2 (k5(1: 210), :); % Dividing the dataset2 into 210
training rows
Data2_testing5 = Data2(k5(211: end), :); % Dividing the dataset2 into 100
test rows

Data2_training_mat5 = table2array(Data2_training5(:,1:6));
training_class_mat5 = table2array(Data2_training5(:,7));
Data2_testing_mat5 = table2array(Data2_testing5(:,1:6));
testing_class_mat5 = table2array(Data2_testing5(:,7));

svmStruct5 = svmtrain(Data2_training_mat5 ,training_class_mat5,
'kernel_function', 'rbf');
groups5 = svmclassify(svmStruct5, Data2_testing_mat5);

%Calculating the confusion matrix for reference
%C = confusionmat(testing_class_mat,groups);

TP5 = sum(groups5 == "Abnormal" & testing_class_mat5=="Normal");
FP5 = sum(groups5 == "Normal" & testing_class_mat5=="Abnormal");
TN5 = sum(groups5 == "Abnormal" & testing_class_mat5=="Abnormal");
FN5 = sum(groups5 == "Abnormal" & testing_class_mat5=="Normal");

```

Below is the code for computing the average number of True positive/negative and False psotive/negative

```

%Computing average of the True/False positive and True/False negative
Avg_TP=(TP1+TP2+TP3+TP4+TP5)/5;
Avg_FP=(FP1+FP2+FP3+FP4+FP5)/5;
Avg_TN=(TN1+TN2+TN3+TN4+TN5)/5;
Avg_FN=(FN1+FN2+FN3+FN4+FN5)/5;

```

Below is the code for computing the confusion matrix

```

%Creating confusion matrix based on the average performance values
C= [Avg_TN Avg_FP; Avg_FN Avg_TP];

```

The below figure represents the confusion matrix.

	1	2
1	58.4000	10.8000
2	4.6000	4.6000



## 2c. Compute the accuracy, precision and recall values.

Solution:

I have calculated the accuracy, precision, and recall using the formula based on the confusion matrix.

```
accuracy=(C(1,1)+C(2,2))/(C(1,1)+C(1,2)+C(2,1)+C(2,2));  
precision=C(2,2)/(C(2,2)+C(1,2));  
recall=(C(2,2)/(C(2,2)+C(2,1)));
```

```
accuracy =
```

```
0.8036
```

```
>> precision
```

```
precision =
```

```
0.2987
```

```
>> recall
```

```
recall =
```

```
0.5000
```

**3. Compare the performance values obtained in 1c and 2c above with the ones you received for the same data in Problem 1(c) of Homework#1. Comment on the differences and you observe and their possible causes/consequences.**

The below screenshot is the performance parameters for 1c:

```
accuracy =  
    0.7692  
  
>> precision  
  
precision =  
    0.1667  
  
>> recall  
  
recall =  
    0.5000
```

The below is the screen shot of the performance parameters for 2c.

```
accuracy =  
    0.8036  
  
>> precision  
  
precision =  
    0.2987  
  
>> recall  
  
recall =  
    0.5000
```

The below is the screen shot for the performance parameters of the Decision Tree's.

Avg_accuracy	0.8905
Avg_precision	0.8435
Avg_recall	0.8202

Accuracy:

I have observed that the accuracy for SVM with kernel function linear has accuracy of 0.7692 whereas in the case of SVM with rbf as a linear function we have 0.8036. But the decision trees have a higher accuracy of 0.8905.

Decision Trees>SVM (rbf)>SVM (linear)

Precision:

The Precision for Decision tree is higher than both the SVM's. The SVM with kernel function rbf has more precision value than the SVM with linear kernel function.

Decision Trees>SVM (rbf)>SVM (linear)

Recall:

The Recall value for the Decision Tree is much more than the Recall values of SVM's. Both the SVM's have the same Recall values.

Decision Tree>SVM (linear and rbf)

The reason why Decision Tree perform better than SVM's because of the use of inappropriate kernel for a linear, non-linear dataset. SVM's are expected to give good results when they are used correctly. SVM's also take a long time to train and also there is a chance that the majority of the data might end up as support vectors.

**4. Consider a dataset of 3-D data points as shown here.**

- a. Show every step of the working of the Perceptron Training Algorithm with this data for only the first two epochs. Use (1, 2, 3, 4) as the initial weight vector.
- b. What is the error value,  $J(w)$ , after each epoch?

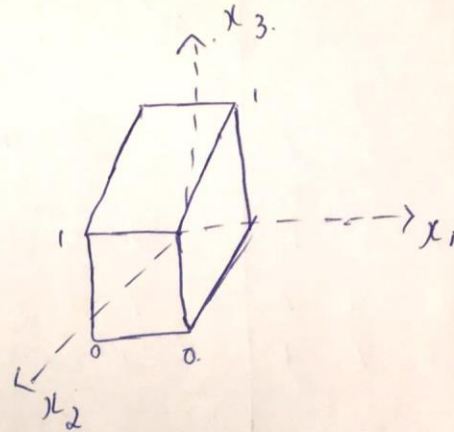
X	Y	Z	Class
5	2	8	1
-2	9	3	0
0	10	4	0
6	1	4	1
8	-3	9	1

Solution:

4.) (i)

$x_1$	$x_2$	$x_3$	Class
5	2	8	1
-2	9	3	0
0	10	4	0
6	1	4	1
8	-3	9	1

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 = 0$$



append  $x_4 = 1$  (Augmented Dataset)

$x_1$	$x_2$	$x_3$	$x_4$	Class
5	2	8	1	1
-2	9	3	1	0
0	10	4	1	0
6	1	4	1	1
8	-3	9	1	1

Selected / initialized weight vector :

1 2 3 4  
 $w_1$   $w_2$   $w_3$   $w_4$

$\vec{x}$				$\vec{w}_t$				$\vec{x} \cdot \vec{w}$	error?	$(\vec{w}_t + x)$
$x_1$	$x_2$	$x_3$	$x_4$	$w_1$	$w_2$	$w_3$	$w_4$		$J(w)$	
5	2	8	1	1	2	3	4	37	N	1 2 3 4
-2	9	3	1	1	2	3	4	29	Y	3 -7 0 3
0	10	4	1	3	-7	0	3	-67	N	3 -7 0 3
6	1	4	1	3	-7	0	3	+14	N	3 -7 0 3
8	-3	9	1	3	-7	0	3	+48	N	3 -7 0 3
5	2	8	1	3	-7	0	3	4	N	3 -7 0 3
-2	9	3	1	3	-7	0	3	-66	N	3 -7 0 3

ALL OK

The final weights <sup>vector</sup> are:

$$[3 \quad -7 \quad 0 \quad 3]$$

$3x_1 - 7x_2 + 0x_3 + 3 = 0$  in the hyperplane.

(ii) is the number of misclassified training vector only one which is  $[-2 \ 9 \ 3 \ 1]$   
So, for each epoch

$$J(w) = \sum_{x \in Y} (\delta_x w^T x)$$

where,  $Y$  is the set of data

&  $\delta_x$  is +1 on the misclassified point belongs to negative class.

$$\therefore J(w) = 1[1 \ 2 \ 3 \ 4] \begin{bmatrix} -2 \\ 9 \\ 3 \\ 1 \end{bmatrix} = -2 + 18 + 9 + 4 = 29$$

$\therefore$  For 1<sup>st</sup> epoch in  $J(w) = 29$ .

For the second epoch,  
As there were no misclassified,  
 $J(w) = 0$ .