CSI 747 – Midterm Submission

Jayasurya Kanukurthy jkanukur@gmu.edu G00863457

Introduction:

The midterm required implementing Support Vector Machine with primal soft-margin method, dual-soft margin method and a dual soft-margin method with a radial basis kernel. This document mentions the process used to prepare data, models implemented in AMPL to train the models; and to test the results obtained from training data.

The following sections discuss each of these implementations – Training and Testing - in more detail:

1. Preparing Data for questions (2), (3), and (4), and Preparing Data for question (5).
2. Implementation of Primal soft-margin SVM to differentiate 3s and 6s.
3. Implementing Dual soft-margin SVM to differentiate 3s and 6s.
4. Dual soft-margin SVM using a Radial Basis Kernel.
5. Dual soft-margin SVM using a Radial Basis Kernel to differentiate even numbers from odd numbers.

Preparing Data Sets:

The data for questions (2),(3) and (4) was taken from files “train3.txt”,”train6.txt” for training the SVM, and “test3.txt”,”test6.txt” for Testing the values obtained from SVM.

500 vectors from each of the above mentioned files were used for forming the training and test set.

These vectors were each normalized before running through the SVM.

The code - implemented in MATLAB - used for forming datasets for (2),(3) and (4) is given below:

Form_3and6_DataSets.m

```
clear all; clc;

train3 = dlmread('midtermdata/train3.txt');

train3_subset = train3(1:500,:);

clear train3;


train6 = dlmread('midtermdata/train6.txt');

train6_subset = train6(1:500,:);

clear train6;
```

```
test3 = dlmread('midtermdata/test3.txt');

test3_subset = test3(1:500,:);

clear test3;

test6 = dlmread('midtermdata/test6.txt');

test6_subset = test6(1:500,:);

clear test6;



train_set_20 = [train3_subset;train6_subset];
test_set_20 = [test3_subset;test6_subset];

for i = 2:length(train_set_20(:,1))
    dr = 0;
    if(max(train_set_20(i,2:end))==0)
        dr=1;
    else
        dr = max(train_set_20(i,2:end));
    end
    train_set_20(i,2:end) = train_set_20(i,2:end)/dr;
end

for i = 2:length(test_set_20(:,1))
    dr = 0;
    if(max(test_set_20(i,2:end))==0)
        dr=1;
    else
        dr = max(test_set_20(i,2:end));
    end
    test_set_20(i,2:end) = test_set_20(i,2:end)/dr;
end



horz = 1:length(train_set_20(1,:));
vert = 0:length(train_set_20(:,1));
train_set_20 = [horz;train_set_20];
train_set_20 = [vert',train_set_20];

horz = 1:length(test_set_20(1,:));
vert = 0:length(test_set_20(:,1));
test_set_20 = [horz;test_set_20];
test_set_20 = [vert',test_set_20];

dlmwrite('3and6_Train_set.txt',train_set_20,' ');
dlmwrite('3and6_Test_set.txt',test_set_20,' ');
```

The Dataset for question 5 was formed by taking 100 first data points from each of the Training and Test datasets.

The code in MATLAB used for forming them is below:

Create_Full_DataSet.m

```
clear all; clc;

display 'creating subset of files';
display 'This might take some time!!';

files = dir('midtermdata/train*.txt');
train = [];
data_set=[];
%Read in RAW data from files
for nm = 1:length(files)
    train_data = dlmread(strcat('midtermdata/',files(nm).name));
    data_set = train_data(1:100,2:end);
    train = [train;data_set];

end
for i = 2:length(train(1,:))
    dr = 0;
    if(max(train(:,i))==0)
        dr = 1;
    else
        dr = max(train(:,i));
    end
    train(:,i) = train(:,i)/dr;
end
horz = 1:length(train(1,:));
vert = 0:length(train(:,1));
train = [horz;train];
train = [vert',train];

dlmwrite('Train_Set_Full.txt',train,' ');
```

Implementing Primal Soft-Margin SVM:

The Primal Soft-Margin SVM was trained using the following AMPL code:

SVM-2.mod: include SVM-2-Train.mod

```
param n := 785;
param D := 1000;
param C := 100;

set POINTS := {2..n};
#set POINTS1 := {1..n-1};
set INPUT_POINTS := {1..n};
set DATASET := {1..D};

param x{DATASET,POINTS1};
param y{DATASET};

data 3and6_Train_set.txt;

Setting the value of y: 1 if digit is 3 and -1 for 6
for {i in DATASET} {

      if x[i,1] = 3 then {let y[i] := 1} else{ let y[i] := -1};

}

var w{POINTS};
var b;
var eta{DATASET} >= 0;

minimize SVM {j in POINTS} : 0.5*(w[j]*w[j])+(C*sum {i in DATASET} eta[i]);

s.t. CONDITION {i in DATASET}: y[i]*( (sum {j in POINTS} (x[i,j]*w[j]))-b) >= 1-
eta[i];

option solver snopt;
solve;
```

The code was run, and gave the following values of w and b:

Result: SVM-2-Train.mod

```
SNOPT 7.2-8 : Optimal solution found.
7500 iterations, objective 0.9331388318

w [*] :=
  2  0            198  0            394  0            590  0
  3  0            199  0            395  0            591  0
  4  0            200  0.00632913   396  0            592  0.00245573
  5  0            201  0.00476042   397  0            593  0.0326093
  6  0            202  0.0206777    398 -0.0125816    594  0.0533608
```

7	0	203	0.0268085	399	-0.0467392	595	0.0389315
8	0	204	0.0488274	400	0.0187036	596	0.034354
9	0	205	0.0389903	401	-0.0270946	597	0.0242443
10	0	206	0.0146756	402	-0.057874	598	0.0181352
11	0	207	0.058661	403	-0.0719348	599	-0.06953
12	0	208	0.0150024	404	-0.0270767	600	-0.0228797
13	0	209	0.0560007	405	-0.0164912	601	0.0171081
14	0	210	0.00505681	406	-0.0962813	602	-0.0804761
15	0	211	0.0220115	407	-0.0390875	603	-0.0980588
16	0	212	0.00560537	408	0.0305464	604	0.00454695
17	0	213	0.0342548	409	0.0805546	605	0.0279356
18	0	214	0.110073	410	0.0147909	606	0.0222861
19	0	215	0.0251302	411	-0.0132871	607	-0.041661
20	0	216	0.0641945	412	0.0957558	608	0.001856
21	0	217	0.0791721	413	0.0190721	609	0.0339414
22	0	218	0.0664382	414	-0.0175872	610	0.0873956
23	0	219	0.0400187	415	-0.0605229	611	0.0320942
24	0	220	0.0294737	416	-0.105098	612	-0.0156752
25	0	221	0.0350157	417	-0.0715644	613	0.000729338
26	0	222	-0.0145561	418	-0.0579831	614	0
27	0	223	-0.0145561	419	-0.00949523	615	0
28	0	224	-0.0145561	420	0	616	0
29	0	225	-0.0145561	421	0	617	0
30	0	226	0	422	0	618	0
31	0	227	0	423	0	619	0
32	0	228	-2.04021e-16	424	0.0173691	620	-8.71776e-17
33	0	229	0.0314506	425	0.0155242	621	0.000265927
34	0	230	0.0404709	426	0.00191582	622	0.0169571
35	0	231	0.0263637	427	-0.0382019	623	0.0446649
36	0	232	0.0270541	428	-0.0246455	624	0.057553
37	8.79568e-17	233	0.00422765	429	-0.0751222	625	0.0146861
38	-2.84025e-17	234	0.020858	430	-0.119963	626	-0.00870476
39	-7.9729e-18	235	0.0353612	431	-0.133554	627	0.0317492
40	6.67412e-14	236	0.0147125	432	-0.0382567	628	-0.0567996
41	-1.81259e-14	237	0.0594455	433	-0.00180544	629	0.066188
42	-6.15046e-17	238	0.004127	434	0.00770934	630	0.00800053
43	-9.73425e-17	239	0.0216688	435	0.0218571	631	-0.0207817
44	-8.26129e-17	240	0.0964585	436	0.0101784	632	0.0154795
45	2.37239e-17	241	0.09235	437	0.0581184	633	0.0837469
46	9.25021e-17	242	0.132497	438	0.0838373	634	0.0316464
47	5.13718e-16	243	0.130264	439	-0.0231305	635	0.0368887
48	-9.47724e-17	244	0.187343	440	0.00897118	636	0.0652766
49	5.57601e-14	245	0.0938227	441	0.0645473	637	0.0579556
50	1.42974e-17	246	0.0340535	442	-0.0135053	638	0.0297489
51	0	247	0.0346553	443	-0.0525532	639	0.0143435
52	0	248	0.0346546	444	-0.0653997	640	7.69512e-05
53	0	249	0.0234793	445	-0.0809408	641	-2.0229e-16
54	0	250	-0.0145561	446	-0.0495833	642	0
55	0	251	0	447	-0.00957026	643	0
56	0	252	-0.0145561	448	0	644	0
57	0	253	0	449	0	645	0
58	0	254	0	450	0	646	0
59	0	255	0	451	0	647	0
60	0	256	0	452	0.0218577	648	0
61	0	257	0.0411285	453	0.0308292	649	1.28812e-16

62	-4.17697e-16	258	0.0164808	454	0.0142416	650	0.016505
63	0	259	-0.00142818	455	-0.0353069	651	0.016505
64	0	260	0.0309659	456	-0.0561341	652	0.0321404
65	-5.39625e-18	261	0.00297123	457	-0.0818247	653	0.0498866
66	-0.000259386	262	-0.0404722	458	-0.184264	654	0.0552521
67	-0.0581369	263	-0.0438159	459	-0.189448	655	0.0384531
68	-0.0182869	264	-0.0137555	460	-0.052322	656	0.112813
69	-0.0178031	265	-0.0442771	461	-0.1617	657	0.0750474
70	-0.0404462	266	-0.013297	462	-0.0501282	658	0.0506983
71	-0.0246079	267	0.0268331	463	-0.0200902	659	0.0570828
72	-0.0430801	268	0.0674106	464	-0.0361804	660	0.0238603
73	-0.0131109	269	0.0878871	465	0.038485	661	-0.0218723
74	-0.00330565	270	0.131054	466	-0.023971	662	-0.0297571
75	-0.00797557	271	0.177812	467	-0.0949299	663	0.0424288
76	-0.00256738	272	0.129346	468	-0.00813703	664	0.0572579
77	-0.00161197	273	0.0692183	469	0.0817487	665	0.0391762
78	-0.000284464	274	-0.011698	470	0.00972408	666	0.0174789
79	-1.39195e-16	275	0.0186208	471	-0.0100166	667	0.0203007
80	-3.13098e-16	276	-0.0218448	472	-0.0445286	668	0.000517747
81	2.16712e-14	277	-0.0278175	473	-0.0974997	669	-1.16825e-14
82	-1.75869e-14	278	-0.020362	474	-0.0399522	670	0
83	0	279	0	475	-0.00957026	671	0
84	0	280	0	476	0	672	0
85	0	281	0	477	0	673	0
86	0	282	0	478	0	674	0
87	0	283	0	479	0	675	0
88	0	284	0	480	0.0270183	676	0
89	0	285	0.0203607	481	0.0368592	677	-1.38182e-16
90	7.26285e-18	286	0.00601203	482	0.0359601	678	0.036598
91	-2.06808e-14	287	0.00754782	483	-0.0157489	679	0.0364538
92	-2.17851e-16	288	0.038933	484	-0.0427694	680	0.0366305
93	-0.0199014	289	-0.0100513	485	-0.070064	681	0.0369158
94	-0.0266273	290	-0.0409634	486	-0.197376	682	0.0360536
95	-0.0578059	291	-0.0395043	487	-0.131679	683	0.0242847
96	-0.0243646	292	-0.0175868	488	-0.178531	684	0.0679241
97	-0.00155791	293	-0.0222605	489	-0.320454	685	0.117503
98	-0.0286154	294	0.0136122	490	-0.186531	686	0.0968808
99	-0.0543864	295	-0.00412512	491	-0.111723	687	0.103679
100	-0.0550355	296	-0.00923371	492	-0.113189	688	0.0957395
101	-0.0266202	297	0.0995732	493	0.0474106	689	0.0484958
102	-0.00184472	298	0.134602	494	-0.0331465	690	0.0642631
103	-0.0335711	299	0.150611	495	-0.00656597	691	0.0714345
104	-0.0523784	300	0.154909	496	-0.00435802	692	0.0699319
105	-0.0437697	301	0.0910238	497	-0.0440807	693	0.0266299
106	-0.0225182	302	0.0202437	498	-0.0598644	694	0.0252161
107	-0.0412614	303	0.0191089	499	0.0370107	695	0.00418634
108	-0.0433415	304	-0.0151083	500	-0.00899237	696	-1.00522e-19
109	-0.0247767	305	-0.0642713	501	-0.0910117	697	0
110	0	306	-0.0220941	502	-0.0112555	698	0
111	0	307	0	503	-0.00884646	699	0
112	0	308	0	504	0	700	0
113	0	309	0	505	0	701	0
114	0	310	0	506	0	702	0
115	0	311	0	507	0	703	0
116	0	312	0	508	0.0184398	704	0

117	0	313	0.0501345	509	0.0340338	705	7.46019e-17
118	-1.53468e-14	314	0.0369451	510	0.0449696	706	0.0245075
119	-0.00943616	315	0.00874006	511	0.0123433	707	0.0215537
120	-0.0107512	316	0.0230115	512	0.0160041	708	0.0223313
121	0.0108339	317	0.0082077	513	0.000856405	709	0.0188472
122	0.0197792	318	0.043257	514	-0.118481	710	0.00821736
123	0.023103	319	-0.0197231	515	-0.134259	711	0.00307238
124	0.0401592	320	-0.0223805	516	-0.179158	712	0.00343781
125	0.0396724	321	0.058326	517	-0.202646	713	0.00553355
126	0.0897961	322	-0.0234288	518	-0.1173	714	0.0252321
127	0.020252	323	0.00660639	519	-0.0972166	715	0.067262
128	-0.0779316	324	0.10659	520	-0.0572413	716	0.0435302
129	-0.0344054	325	0.144185	521	-6.06309e-06	717	0.0134131
130	0.0370221	326	0.0459134	522	-0.0395031	718	0.00651994
131	0.0217565	327	0.141163	523	-0.010108	719	-5.20972e-17
132	0.0196177	328	0.16169	524	-0.028028	720	0.00282762
133	-0.133722	329	0.120535	525	-0.064507	721	0.0078615
134	-0.179007	330	0.03498	526	-0.0236902	722	0.0078615
135	-0.111586	331	0.00808642	527	-0.000111939	723	0.0078615
136	-0.0531318	332	0.0132422	528	0.0360508	724	0
137	-0.0170078	333	-0.0680615	529	-0.0566487	725	0
138	6.44398e-14	334	-0.0501042	530	-0.00942532	726	0
139	0	335	3.271e-14	531	-0.00569469	727	0
140	0	336	0	532	0	728	0
141	0	337	0	533	0	729	0
142	0	338	0	534	0	730	0
143	0	339	0	535	0	731	0
144	0	340	0	536	0.00464161	732	0
145	-1.15368e-16	341	0.0436595	537	0.0159155	733	0
146	-0.00492316	342	0.030906	538	0.0134842	734	0
147	-0.0121125	343	0.00546381	539	0.0123674	735	0
148	0.000705554	344	0.0285102	540	0.0396679	736	0
149	0.0793507	345	0.0645023	541	0.0770845	737	0
150	0.0551969	346	0.0337041	542	-0.0141297	738	0
151	0.0266743	347	-0.02431	543	-0.110427	739	0
152	0.0489251	348	-0.0152063	544	-0.129196	740	0.0116178
153	0.0582853	349	0.0230228	545	-0.180832	741	0.0116178
154	0.0598178	350	-0.0408621	546	-0.111158	742	0.0116178
155	0.0818831	351	-0.00135278	547	-0.0779778	743	0.0116178
156	-0.0126922	352	0.09315	548	-0.0606605	744	0.0116178
157	0.0545111	353	0.0355177	549	-0.0624464	745	0.0116178
158	0.0850526	354	0.0123443	550	0.00801447	746	0
159	-0.0386638	355	0.0305722	551	0.00708408	747	0
160	-0.0319913	356	0.0619872	552	0.0544045	748	0
161	-0.109335	357	0.0650978	553	0.0175076	749	0
162	-0.132979	358	0.00981814	554	-0.0185216	750	0
163	-0.138792	359	-0.0494686	555	-0.0339329	751	0
164	-0.080498	360	-0.040193	556	0.0799483	752	0
165	-0.0134477	361	-0.0504729	557	-0.0171499	753	0
166	-0.013419	362	-0.0699581	558	-0.00957026	754	0
167	-0.0145561	363	1.16723e-17	559	-0.00683594	755	0
168	-0.0145561	364	0	560	0	756	0
169	0	365	0	561	0	757	0
170	0	366	0	562	0	758	0
171	0	367	0	563	0	759	0

```
172  0.0078615     368  0             564  0.00727853    760  0
173  0.00702378    369  0.0436595     565  0.0348105     761  0
174 -0.000951051   370  0.0259214     566  0.0170382     762  0
175 -0.0102713     371  0.000171597   567  0.00389667    763  0
176  0.0242834     372  0.0390645     568  0.0244985     764  0
177  0.0742141     373  0.056723      569  0.0695047     765  0
178 -0.0105544     374  0.015285      570  0.0343081     766  0
179 -0.0208608     375 -0.0472017     571 -0.0658294     767  0
180  0.0134156     376 -0.0398539     572 -0.0483423     768  0
181  0.0505063     377 -0.00882352    573 -0.0444746     769  0
182  0.0653217     378 -0.097911      574 -0.0633916     770  0
183  0.0918342     379 -0.0533587     575  0.00575063    771  0
184  0.0123267     380  0.0346085     576 -0.00630118    772  0
185 -0.0149629     381  0.0329156     577 -0.064012      773  0
186  0.0743554     382 -0.00408859    578 -0.0358787     774  0
187  0.00388012    383  0.071261      579 -0.0300458     775  0
188  0.0396797     384  0.0875986     580  0.00311627    776  0
189 -0.028281      385  0.00465692    581 -0.0209204     777  0
190  0.0275798     386 -0.00912711    582 -0.0628325     778  0
191  0.0219908     387 -0.0826662     583 -0.0105376     779  0
192 -0.0333734     388 -0.0783939     584  0.0341486     780  0
193  0.00669146    389 -0.0564344     585 -0.00620682    781  0
194 -0.0145561     390 -0.0650109     586 -0.000743322   782  0
195 -0.0145561     391 -0.00629943    587  2.28871e-14   783  0
196 -0.0145561     392  0             588  0             784  0
197 -0.0145561     393  0             589  0             785  0
;

b = -0.100834
```

These values of W and b were used on the Test Dataset to determine if the digit was 3 or 6. The model for Testing was implemented as follows:

SVM-2.mod: include SVM-2-Test.mod

```
param x_test{DATASET,INPUT_POINTS};
param y_test{DATASET};

param test{DATASET};

param hit3; param miss3;
param hit6; param miss6;

param total_hits; param total_misses;


data 3and6_Test_set.txt;

let hit3 := 0; let miss3 := 0;
let hit6:= 0; let miss6 := 0;
let total_hits :=0; let total_misses:=0;
```

```
#Checking number of misses for digit 3
for {i in {1..500}} {

	#Finding where the point is placed on the hyperplane
	let test[i] := sum {j in POINTS} (x_test[i,j]*w[j]) - b;
	if (test[i] >= 0) then
	{
		let y_test[i] := 1;
		let hit3 := hit3+1
	}
	else
	{
		let y_test[i] := -1;
		let miss3 := miss3+1
	};
};

#Checking number of misses for digit 6
for {i in {501..1000}} {

	#Finding where the point is placed on the hyperplane
	let test[i] := sum {j in POINTS} (x_test[i,j]*w[j]) - b;
	if (test[i] >= 0) then
	{
		let y_test[i] := 1;
		let miss6 := miss6+1
	}
	else
	{
		let y_test[i] := -1;
		let hit6 := hit6+1
	};
};

#Calculating Accuracy and Errors
for {i in DATASET}
{
	if(test[i] >= 0 and x_test[i,1] == 3) then
	{
		let total_hits:=total_hits+1;
	}
	else if(test[i] < 0 and x_test[i,1] == 6) then
	{
		let total_hits := total_hits+1;
	}
	else
	{
		let total_misses := total_misses + 1;
	};
};

display y_test;

display hit3, miss3, hit6, miss6;
```

```
display total_hits, total_misses;
```

The output generated by testing on the Test Data set was as follows:

SVM-2-Test.mod: Output

```
y_test [*] :=
   1  1    126  1    251  1    376  1    501 -1    626 -1    751 -1    876 -1
   2  1    127  1    252  1    377  1    502 -1    627 -1    752 -1    877 -1
   3  1    128  1    253  1    378  1    503 -1    628 -1    753 -1    878 -1
   4  1    129  1    254  1    379  1    504  1    629 -1    754 -1    879 -1
   5  1    130  1    255  1    380  1    505 -1    630 -1    755 -1    880 -1
   6  1    131  1    256  1    381  1    506 -1    631 -1    756 -1    881 -1
   7  1    132  1    257  1    382  1    507 -1    632 -1    757 -1    882 -1
   8  1    133  1    258  1    383  1    508 -1    633 -1    758 -1    883 -1
   9  1    134  1    259  1    384  1    509 -1    634 -1    759 -1    884 -1
  10  1    135  1    260  1    385  1    510 -1    635 -1    760 -1    885 -1
  11  1    136  1    261  1    386  1    511 -1    636 -1    761 -1    886 -1
  12  1    137  1    262  1    387  1    512 -1    637 -1    762 -1    887 -1
  13  1    138  1    263  1    388  1    513 -1    638 -1    763 -1    888 -1
  14  1    139  1    264  1    389  1    514 -1    639 -1    764 -1    889 -1
  15  1    140  1    265  1    390  1    515 -1    640 -1    765 -1    890 -1
  16  1    141  1    266  1    391  1    516 -1    641 -1    766 -1    891 -1
  17  1    142  1    267  1    392  1    517 -1    642 -1    767 -1    892 -1
  18  1    143  1    268  1    393  1    518 -1    643 -1    768 -1    893 -1
  19  1    144  1    269  1    394  1    519 -1    644 -1    769 -1    894 -1
  20  1    145  1    270  1    395  1    520 -1    645 -1    770 -1    895 -1
  21  1    146  1    271  1    396  1    521 -1    646 -1    771 -1    896 -1
  22  1    147  1    272  1    397  1    522 -1    647 -1    772 -1    897 -1
  23  1    148  1    273  1    398  1    523 -1    648 -1    773 -1    898 -1
  24  1    149  1    274  1    399  1    524 -1    649 -1    774 -1    899 -1
  25  1    150  1    275  1    400  1    525 -1    650 -1    775 -1    900 -1
  26  1    151  1    276  1    401  1    526 -1    651 -1    776 -1    901 -1
  27  1    152  1    277  1    402  1    527 -1    652 -1    777 -1    902 -1
  28  1    153  1    278  1    403  1    528 -1    653 -1    778 -1    903 -1
  29  1    154  1    279  1    404  1    529 -1    654 -1    779 -1    904 -1
  30  1    155  1    280  1    405  1    530 -1    655 -1    780 -1    905 -1
  31  1    156  1    281  1    406  1    531 -1    656 -1    781 -1    906 -1
  32  1    157  1    282  1    407  1    532 -1    657 -1    782 -1    907 -1
  33  1    158  1    283  1    408  1    533 -1    658 -1    783 -1    908 -1
  34  1    159  1    284  1    409  1    534 -1    659 -1    784 -1    909 -1
  35  1    160  1    285  1    410  1    535 -1    660 -1    785 -1    910 -1
  36 -1    161  1    286  1    411  1    536 -1    661 -1    786 -1    911 -1
  37  1    162  1    287  1    412  1    537 -1    662 -1    787 -1    912 -1
  38  1    163  1    288  1    413  1    538 -1    663 -1    788 -1    913 -1
  39  1    164  1    289  1    414  1    539 -1    664 -1    789 -1    914 -1
  40  1    165  1    290  1    415  1    540 -1    665 -1    790 -1    915 -1
  41  1    166  1    291  1    416  1    541 -1    666 -1    791 -1    916 -1
  42  1    167 -1    292  1    417  1    542 -1    667 -1    792 -1    917 -1
  43  1    168  1    293  1    418  1    543 -1    668  1    793 -1    918 -1
  44  1    169  1    294  1    419  1    544 -1    669 -1    794 -1    919 -1
  45  1    170  1    295  1    420  1    545 -1    670 -1    795 -1    920 -1
  46  1    171  1    296  1    421  1    546 -1    671 -1    796 -1    921 -1
  47  1    172  1    297  1    422  1    547 -1    672 -1    797 -1    922 -1
```

48	1	173	1	298	1	423	1	548	-1	673	-1	798	-1	923	-1
49	1	174	1	299	1	424	1	549	-1	674	-1	799	-1	924	-1
50	1	175	1	300	1	425	1	550	-1	675	-1	800	-1	925	-1
51	1	176	1	301	1	426	1	551	-1	676	-1	801	-1	926	-1
52	1	177	1	302	1	427	1	552	-1	677	-1	802	-1	927	-1
53	1	178	1	303	1	428	1	553	-1	678	-1	803	-1	928	-1
54	1	179	1	304	1	429	1	554	-1	679	-1	804	-1	929	-1
55	1	180	1	305	1	430	1	555	-1	680	-1	805	-1	930	-1
56	1	181	1	306	-1	431	1	556	1	681	-1	806	-1	931	-1
57	1	182	1	307	1	432	1	557	-1	682	-1	807	-1	932	-1
58	1	183	1	308	1	433	1	558	-1	683	-1	808	-1	933	-1
59	1	184	1	309	1	434	1	559	-1	684	-1	809	-1	934	-1
60	1	185	1	310	-1	435	1	560	-1	685	-1	810	-1	935	-1
61	1	186	1	311	1	436	1	561	-1	686	-1	811	-1	936	-1
62	1	187	1	312	1	437	1	562	-1	687	-1	812	-1	937	-1
63	1	188	1	313	-1	438	1	563	-1	688	-1	813	-1	938	-1
64	1	189	1	314	-1	439	1	564	-1	689	-1	814	-1	939	-1
65	1	190	1	315	1	440	1	565	-1	690	-1	815	-1	940	-1
66	1	191	1	316	1	441	1	566	-1	691	-1	816	-1	941	-1
67	1	192	1	317	1	442	1	567	-1	692	-1	817	-1	942	-1
68	1	193	1	318	1	443	1	568	-1	693	-1	818	-1	943	-1
69	1	194	-1	319	1	444	1	569	-1	694	-1	819	-1	944	-1
70	1	195	1	320	1	445	1	570	-1	695	-1	820	-1	945	-1
71	1	196	1	321	1	446	1	571	-1	696	-1	821	-1	946	-1
72	1	197	1	322	1	447	1	572	-1	697	-1	822	-1	947	-1
73	1	198	1	323	1	448	1	573	-1	698	-1	823	-1	948	-1
74	1	199	1	324	1	449	1	574	-1	699	-1	824	-1	949	-1
75	1	200	1	325	1	450	1	575	-1	700	-1	825	-1	950	-1
76	1	201	1	326	1	451	1	576	-1	701	-1	826	-1	951	-1
77	1	202	1	327	1	452	1	577	-1	702	-1	827	-1	952	-1
78	1	203	1	328	1	453	1	578	-1	703	-1	828	-1	953	-1
79	1	204	1	329	1	454	1	579	-1	704	-1	829	-1	954	-1
80	1	205	1	330	1	455	1	580	-1	705	-1	830	-1	955	-1
81	1	206	1	331	1	456	1	581	-1	706	-1	831	-1	956	-1
82	1	207	1	332	1	457	1	582	-1	707	-1	832	-1	957	-1
83	1	208	1	333	1	458	1	583	-1	708	-1	833	-1	958	-1
84	1	209	1	334	1	459	1	584	-1	709	-1	834	-1	959	-1
85	1	210	1	335	1	460	1	585	-1	710	-1	835	-1	960	-1
86	1	211	1	336	1	461	1	586	-1	711	-1	836	-1	961	-1
87	1	212	1	337	1	462	1	587	-1	712	-1	837	-1	962	-1
88	1	213	1	338	1	463	1	588	-1	713	-1	838	-1	963	-1
89	1	214	1	339	1	464	1	589	-1	714	-1	839	-1	964	-1
90	1	215	1	340	1	465	1	590	-1	715	-1	840	-1	965	-1
91	1	216	1	341	1	466	1	591	-1	716	-1	841	-1	966	-1
92	1	217	1	342	1	467	1	592	-1	717	-1	842	-1	967	-1
93	1	218	1	343	1	468	1	593	-1	718	-1	843	-1	968	-1
94	1	219	1	344	1	469	1	594	-1	719	-1	844	-1	969	-1
95	1	220	1	345	1	470	1	595	-1	720	-1	845	-1	970	-1
96	1	221	1	346	1	471	1	596	-1	721	-1	846	-1	971	-1
97	1	222	1	347	1	472	1	597	-1	722	-1	847	-1	972	-1
98	1	223	1	348	1	473	1	598	-1	723	-1	848	-1	973	-1
99	1	224	1	349	1	474	1	599	-1	724	1	849	-1	974	-1
100	1	225	1	350	1	475	1	600	-1	725	-1	850	-1	975	-1
101	1	226	1	351	1	476	1	601	-1	726	-1	851	-1	976	-1
102	1	227	1	352	1	477	1	602	-1	727	-1	852	-1	977	-1

```
 103  1    228  1    353  1    478  1    603 -1    728 -1    853 -1    978 -1
 104  1    229  1    354  1    479  1    604 -1    729 -1    854 -1    979 -1
 105  1    230  1    355  1    480  1    605 -1    730 -1    855 -1    980 -1
 106  1    231  1    356  1    481  1    606 -1    731 -1    856 -1    981 -1
 107  1    232  1    357  1    482  1    607 -1    732 -1    857 -1    982 -1
 108  1    233 -1    358  1    483  1    608 -1    733 -1    858 -1    983 -1
 109  1    234  1    359  1    484  1    609 -1    734 -1    859 -1    984 -1
 110  1    235  1    360  1    485  1    610 -1    735 -1    860 -1    985 -1
 111  1    236  1    361  1    486  1    611 -1    736 -1    861 -1    986 -1
 112  1    237  1    362  1    487  1    612 -1    737 -1    862 -1    987 -1
 113  1    238  1    363  1    488  1    613 -1    738 -1    863 -1    988 -1
 114  1    239  1    364  1    489  1    614 -1    739 -1    864 -1    989 -1
 115  1    240  1    365  1    490  1    615 -1    740 -1    865 -1    990 -1
 116  1    241  1    366  1    491 -1    616 -1    741 -1    866 -1    991 -1
 117  1    242  1    367  1    492 -1    617 -1    742 -1    867 -1    992 -1
 118  1    243  1    368  1    493  1    618 -1    743 -1    868 -1    993 -1
 119  1    244  1    369  1    494  1    619  1    744 -1    869 -1    994 -1
 120  1    245  1    370  1    495  1    620 -1    745 -1    870 -1    995 -1
 121  1    246  1    371  1    496  1    621 -1    746 -1    871 -1    996 -1
 122  1    247  1    372  1    497  1    622 -1    747 -1    872 -1    997 -1
 123  1    248  1    373  1    498  1    623 -1    748 -1    873 -1    998 -1
 124  1    249  1    374  1    499  1    624 -1    749 -1    874 -1    999 -1
 125  1    250  1    375  1    500  1    625 -1    750 -1    875 -1   1000 -1
;

hit3 = 490
miss3 = 10
hit6 = 495
miss6 = 5

total_hits = 985
total_misses = 15

Accuracy_Percent = 98.5

Error_Percent = 1.5
```

Dual-Soft Margin SVM:

Dual-Soft Margin SVM was implemented in AMPL and Tested the values using MATLAB.

The code used for training SVM in AMPL is as follows:

SVM-3.mod

```
param n := 785;
param D := 1000;
param C := 100;

set POINTS := {2..n};
set INPUT_POINTS := {1..n};
set DATASET := {1..D};
```

```
param x{DATASET,INPUT_POINTS};
param y{DATASET};
#param K{DATASET,DATASET};

data 3and6_Train_set.txt;

param K{i in DATASET, j in DATASET} := sum {t in POINTS} x[i,t]*x[j,t];

#Initializing Y[i]
for {i in DATASET} {

      if x[i,1] = 3 then {let y[i] := 1} else{ let y[i] := -1};

}


var alpha{DATASET} >= 0, <= C;   #

maximize DUALSVM : (sum {i in DATASET} alpha[i]) - 0.5*(sum {i in DATASET,j in
DATASET} alpha[i]*alpha[j]*y[i]*y[j]*K[i,j]);

s.t. CONDITION :  sum {i in DATASET} alpha[i]*y[i] = 0;

option solver snopt;
solve;
```

The values for alpha generated by this SVM is as follows:

SVM-3.mod: Output

```
ampl: model SVM-3.mod;
SNOPT 7.2-8 : Optimal solution found.
101 iterations, objective 0.9331388318
ampl: display alpha;
alpha [*] :
   1 0              251 0              501 0              751 0
   2 0              252 0              502 0              752 0
   3 0              253 0              503 0              753 0
   4 0              254 0              504 0              754 0
   5 0              255 0              505 0              755 0
   6 0              256 0              506 0              756 0
   7 0              257 0              507 0              757 0
   8 0              258 0              508 0              758 0
   9 0              259 0              509 0              759 0
  10 0              260 0              510 0              760 0
  11 0              261 0              511 0              761 0
  12 0              262 0              512 0              762 0
  13 0              263 0              513 0              763 0
  14 0              264 0              514 0              764 0
  15 0              265 0              515 0              765 0
  16 0              266 0.00762124     516 0              766 0
  17 0              267 0              517 0              767 0
  18 0              268 0              518 0              768 0
```

19 0.0229393	269 0	519 0.010818	769 0.00438357
20 0	270 0	520 0	770 0
21 0.0158137	271 0	521 0	771 0
22 0	272 0	522 0	772 0
23 0	273 0	523 0	773 0
24 0	274 0	524 0	774 0
25 0	275 0	525 0.0396946	775 0
26 0	276 0.0138238	526 0.00981295	776 0
27 0	277 0	527 0	777 0
28 0	278 0	528 0	778 0
29 0	279 0	529 0	779 0
30 0	280 0.0116178	530 0	780 0
31 0	281 0.018492	531 0	781 0
32 0	282 0	532 0	782 0
33 0	283 0.0643798	533 0	783 0
34 0	284 0	534 0	784 0
35 0	285 0	535 0	785 0
36 0	286 0	536 0	786 0
37 0.036598	287 0	537 0	787 0
38 0	288 0	538 0	788 0
39 0	289 0	539 0.020362	789 0.00778625
40 0	290 0	540 0	790 0.00504809
41 0	291 0	541 0	791 0
42 0	292 0	542 0	792 0
43 0	293 0	543 0	793 0
44 0	294 0	544 0	794 0.0659321
45 0	295 0	545 0	795 0
46 0	296 0	546 0	796 0
47 0	297 0	547 0	797 0
48 0	298 0	548 0	798 0
49 0	299 0	549 0.0330524	799 0
50 0	300 0	550 0	800 0
51 0	301 0	551 0	801 0
52 0	302 0	552 0	802 0
53 0	303 0	553 0	803 0
54 0.00313437	304 0	554 0	804 0
55 0	305 0	555 0	805 0
56 0	306 0	556 0	806 0
57 0	307 0	557 0	807 0
58 0	308 0	558 0	808 0
59 0	309 0	559 0	809 0
60 0	310 0	560 0	810 0.0457639
61 0	311 0	561 0	811 0
62 0	312 0	562 0	812 0
63 0	313 0	563 0	813 0
64 0	314 0	564 0	814 0
65 0	315 0.00878912	565 0.00161197	815 0
66 0	316 0	566 0	816 0
67 0	317 0	567 0	817 0
68 0	318 0.0206537	568 0	818 0
69 0	319 0	569 0	819 0
70 0	320 0	570 0	820 0.0472099
71 0	321 0	571 0	821 0
72 0	322 0	572 0	822 0.0413412
73 0	323 0.0509329	573 0	823 0

74	0	324	0	574	0	824	0
75	0	325	0	575	0	825	0
76	0.0836223	326	0	576	0	826	0
77	0.0171851	327	0	577	0	827	0
78	0	328	0	578	0	828	0
79	0	329	0	579	0	829	0
80	0	330	0	580	0	830	0
81	0	331	0	581	0	831	0
82	0	332	0	582	0	832	0
83	0	333	0	583	0	833	0
84	0	334	0	584	0	834	0
85	0	335	0	585	0	835	0
86	0	336	0	586	0	836	0
87	0	337	0	587	0	837	0
88	0	338	0	588	0	838	0
89	0.00757683	339	0	589	0	839	0
90	0	340	0.0160459	590	0	840	0
91	0	341	0	591	0	841	0
92	0	342	0	592	0	842	0
93	0	343	0.0166185	593	0	843	0
94	0	344	0	594	0	844	0
95	0	345	0	595	0	845	0
96	0	346	0	596	0	846	0
97	0	347	0	597	0	847	0
98	0	348	0	598	0	848	0
99	0	349	0	599	0	849	0
100	0	350	0	600	0.0326473	850	0
101	0	351	0	601	0	851	0
102	0	352	0	602	0	852	0
103	0	353	0	603	0	853	0.042452
104	0	354	0	604	0	854	0
105	0	355	0	605	0	855	0
106	0	356	0	606	0	856	0
107	0	357	0	607	0	857	0
108	0	358	0	608	0	858	0
109	0	359	0	609	0	859	0
110	0	360	0	610	0	860	0
111	0	361	0	611	0.0550584	861	0
112	0	362	0	612	0	862	0.068372
113	0	363	0	613	0	863	0
114	0	364	0	614	0	864	0
115	0	365	0	615	0	865	0
116	0	366	0	616	0	866	0
117	0	367	0	617	0.0178404	867	0
118	0	368	0	618	0	868	0
119	0	369	0	619	0	869	0
120	0	370	0	620	0.0111894	870	0
121	0	371	0	621	0	871	0
122	0	372	0	622	0	872	0
123	0	373	0	623	0	873	0
124	0	374	0	624	0	874	0
125	0.0466636	375	0	625	0	875	0
126	0	376	0	626	0	876	0.0440758
127	0	377	0	627	0	877	0
128	0	378	0	628	0	878	0.0379009

129 0 379 0 629 0 879 0
130 0 380 0 630 0 880 0
131 0.00406832 381 0 631 0 881 0.0125816
132 0 382 0 632 0 882 0
133 0 383 0 633 0 883 0
134 0 384 0 634 0 884 0
135 0 385 0 635 0 885 0
136 0 386 0 636 0 886 0
137 0 387 0 637 0 887 0
138 0 388 0 638 0 888 0
139 0 389 0.0449771 639 0 889 0
140 0 390 0 640 0 890 0
141 0 391 0 641 0 891 0
142 0.0867509 392 0 642 0 892 0
143 0 393 0 643 0 893 0
144 0 394 0 644 0 894 0
145 0 395 0 645 0 895 0
146 0 396 0 646 0 896 0
147 0 397 0 647 0 897 0
148 0 398 0 648 0 898 0
149 0 399 0.0276439 649 0 899 0
150 0 400 0 650 0 900 0
151 0 401 0 651 0 901 0
152 0 402 0.0078615 652 0.012927 902 0
153 0 403 0 653 0 903 0
154 0 404 0 654 0 904 0
155 0 405 0 655 0 905 0
156 0 406 0 656 0.0775996 906 0
157 0 407 0 657 0.00957026 907 0
158 0 408 0.0436595 658 0 908 0
159 0 409 0 659 0 909 0
160 0 410 0 660 0 910 0
161 0 411 0 661 0 911 0
162 0 412 0 662 0 912 0
163 0 413 0 663 0 913 0
164 0 414 0 664 0 914 0
165 0 415 0 665 0 915 0
166 0 416 0 666 0 916 0
167 0 417 0 667 0 917 0
168 0 418 0 668 0 918 0
169 0 419 0 669 0 919 0
170 0 420 0 670 0 920 0
171 0 421 0 671 0.0133647 921 0
172 0 422 0 672 0 922 0
173 0 423 0 673 0 923 0
174 0.0732797 424 0 674 0 924 0
175 0 425 0 675 0 925 0
176 0 426 0 676 0 926 0
177 0 427 0 677 0 927 0
178 0 428 0 678 0 928 0
179 0 429 0 679 0 929 0
180 0 430 0 680 0 930 0
181 0 431 0 681 0 931 0
182 0 432 0 682 0 932 0.0396808
183 0 433 0 683 0 933 0

184 0	434 0	684 0	934 0
185 0	435 0	685 0	935 0
186 0	436 0	686 0	936 0
187 0	437 0	687 0	937 0
188 0.00476878	438 0	688 0	938 0
189 0	439 0	689 0	939 0
190 0.00345133	440 0	690 0	940 0
191 0	441 0	691 0	941 0
192 0	442 0	692 0	942 0
193 0	443 0	693 0	943 0
194 0	444 0	694 0	944 0
195 0	445 0	695 0	945 0
196 0	446 0	696 0	946 0
197 0	447 0	697 0	947 0
198 0	448 0	698 0	948 0
199 0	449 0	699 0	949 0
200 0	450 0	700 0	950 0
201 0	451 0	701 0	951 0
202 0	452 0	702 0	952 0.000771399
203 0	453 0	703 0	953 0
204 0	454 0	704 0	954 0
205 0	455 0	705 0	955 0
206 0	456 0	706 0.00867424	956 0
207 0	457 0	707 0	957 0.0429957
208 0	458 0	708 0	958 0
209 0	459 0	709 0	959 0
210 0	460 0	710 0	960 0
211 0	461 0	711 0	961 0.00962893
212 0	462 0	712 0	962 0
213 0	463 0.0335068	713 0	963 0
214 0	464 0	714 0	964 0
215 0	465 0	715 0.00258128	965 0
216 0.00463581	466 0	716 0	966 0
217 0	467 0	717 0	967 0
218 0	468 0	718 0	968 0
219 0	469 0	719 0	969 0
220 0	470 0	720 0.0145561	970 0
221 0	471 0	721 0	971 0
222 0	472 0	722 0	972 0
223 0	473 0	723 0	973 0
224 0	474 0	724 0	974 0
225 0	475 0	725 0	975 0
226 0	476 0	726 0	976 0
227 0.0548817	477 0	727 0	977 0
228 0	478 0	728 0	978 0
229 0	479 0	729 0.0207381	979 0
230 0	480 0	730 0	980 0
231 0	481 0.0501023	731 0	981 0
232 0	482 0	732 0	982 0
233 0.0107433	483 0	733 0	983 0
234 0	484 0	734 0.0149577	984 0
235 0	485 0	735 0	985 0
236 0	486 0	736 0	986 0
237 0	487 0	737 0	987 0.0101584
238 0	488 0	738 0	988 0

```
 239 0              489 0              739 0              989 0
 240 0              490 0              740 0              990 0
 241 0              491 0              741 0              991 0
 242 0              492 0              742 0              992 0
 243 0              493 0              743 0              993 0
 244 0              494 0              744 0              994 0
 245 0              495 0.00293064     745 0              995 0
 246 0              496 0              746 0              996 0
 247 0              497 0              747 0              997 0
 248 0              498 0              748 0              998 0
 249 0              499 0              749 0              999 0
 250 0              500 0.0173691      750 0             1000 0
;
```

These values of alpha were output into an ASCII file and MATLAB was used to test the SVM output. The code used and the results are as follows:

Test.m : MATLAB code to Test SVM-3.mod output

```
clear all; clc;
ans = dlmread('Train_Set_Matlab.txt');
Train_Set = ans(:,3:end);
ans = dlmread('Test_Set_Matlab.txt');
Test_Set = ans(:,3:end);

op = dlmread('output.txt');
sv = [];
for (i = 1:length(op))
if(op(i,2)>0)
sv = [sv;op(i,1)];
end
end
sv_alpha = op(sv,2);
sv_y = op(sv,3);
support_vectors = [sv,sv_alpha,sv_y];

%Forming the kernel
Kernel = Train_Set*Train_Set';

%Calculating b for each of the vectors to check if all are equal
for(i = 1:length(support_vectors))
    summation = 0;
    for(j = 1:length(Train_Set))
        summation = summation+op(j,2)*op(j,3)*Kernel(j,support_vectors(i,1));
    end
    b = summation - support_vectors(i,3);
    B(i) = b;
end


%Testing if alpha obtained is correct
```

```
for(i = 1:length(Test_Set))
    sum = 0;
    for (j = 1:length(support_vectors))
        sum = sum +
(support_vectors(j,3)*support_vectors(j,2)*(Train_Set(support_vectors(j,1),:)
*Test_Set(i,:)'));
    end
    sum = sum - b;
    if(sum >= 0)
        y_result(i) = 1;
    else
        y_result(i)=-1;
    end
end
y_result=y_result';

hits = 0;
misses = 0;
for (i = 1:length(y_result))
    if(y_result(i) == op(i,3))
        hits = hits+1;
    else
        misses = misses+1;
    end
end
display ('The Value of b for various support vectors:');
B

display ('Checking the Obtained values of Y for accuracy and errors:')
hits
misses

Accuracy_Percent = hits/10
Error_Percent = misses/10
```

The output obtained is as follows:

The Value of b for various support vectors:

B =

Columns 1 through 10

-0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1009 -0.1009 -0.1008 -0.1008 -0.1008

Columns 11 through 20

-0.1009 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1009 -0.1008 -0.1008 -0.1008

Columns 21 through 30

-0.1008 -0.1008 -0.1009 -0.1008 -0.1008 -0.1008 -0.1008 -0.1009 -0.1008 -0.1009

Columns 31 through 40

-0.1008 -0.1008 -0.1008 -0.1009 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008

Columns 41 through 50

-0.1009 -0.1008 -0.1008 -0.1008 -0.1008 -0.1009 -0.1008 -0.1008 -0.1008 -0.1008

Columns 51 through 60

-0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1008

Columns 61 through 70

-0.1008 -0.1008 -0.1008 -0.1008 -0.1008 -0.1009 -0.1008 -0.1008 -0.1008 -0.1008

Checking the Obtained Values for Accuracy:

hits =

985

misses =

15

Accuracy_Percent =

98.5000

Error_Percent =

Dual Soft-Margin SVM Using Radial Basis Kernel:

The training was implemented in AMPL with the following code:

SVM-4.mod

```
reset;
param n := 785;
param D := 1000;
param C := 1000;
param gama := 0.05;

set POINTS := {2..n};
set INPUT_POINTS := {1..n};
set DATASET := {1..D};

param x{DATASET,INPUT_POINTS};
param y{DATASET};

data 3and6_Train_set.txt;

#Initializing KERNEL Function: Radial-basis function machine with K[i,j] = e^(-
g*(x[j]-x[i])^2)
param K{i in DATASET, j in DATASET} := exp (-gama*(sum{t in POINTS} (x[i,t]-
x[j,t])^2));

#Initializing Y[i]
for {i in DATASET} {
```

```
      if x[i,1] = 3 then {let y[i] := 1} else{ let y[i] := -1};

}

var alpha{DATASET} >= 0, <= C;   #

maximize DUALSVM : (sum {i in DATASET} alpha[i]) - 0.5*(sum {i in DATASET,j in
DATASET} alpha[i]*alpha[j]*y[i]*y[j]*K[i,j]);

s.t. CONDITION :  sum {i in DATASET} alpha[i]*y[i] = 0;

option solver snopt;
solve;
```

The output generated by modeling SVM using radial basis function is as follows:

SVM-4.mod: Output

```
ampl: model SVM-4.mod;
SNOPT 7.2-8 : Optimal solution found.
586 iterations, objective 87.76935479
ampl: display alpha;
alpha [*] :=
   1 0            251 0.543351      501 0            751 0.337143
   2 0            252 0.552183      502 0            752 0.280072
   3 0            253 0.0225839     503 0            753 0.369248
   4 0.0828238    254 0             504 0            754 0.457481
   5 0.405296     255 0             505 0.132391     755 0.201131
   6 0            256 0             506 0            756 0.0359257
   7 0            257 0.0870987     507 0.118811     757 0.213857
   8 0.330414     258 0.0457662     508 0            758 0
   9 0            259 0.374511      509 0            759 0
  10 0            260 0.0751795     510 0            760 0.032983
  11 0            261 0.224942      511 0            761 0
  12 0            262 0.32666       512 0.199674     762 0
  13 0            263 0             513 0.370024     763 0.107191
  14 0            264 0.0243264     514 0.590751     764 0
  15 0.216766     265 0             515 0            765 0
  16 0.47841      266 0.408875      516 0            766 0.228641
  17 0.201054     267 0             517 0.239417     767 0.147964
  18 0.408476     268 0.306282      518 0            768 0
  19 0.628396     269 0.471984      519 0.162162     769 0.686097
  20 0            270 0.238273      520 0            770 0
  21 0.877585     271 0             521 0            771 0.0948227
  22 0.345111     272 0             522 0            772 0
  23 0            273 0             523 0            773 0.642589
  24 0            274 0.0582172     524 0            774 0.241383
  25 0            275 0.216708      525 0.83491      775 0.185831
  26 0            276 0.518567      526 0.867771     776 0
  27 0            277 0.0973474     527 0.21089      777 0
  28 0.182475     278 0.209095      528 0.271342     778 0
  29 0.261304     279 0.608632      529 0.363801     779 0
```

30 0	280 0.792564	530 0.251013	780 0.442396
31 0.360436	281 0.441873	531 0.737655	781 0.518337
32 0.135854	282 0.186943	532 0.128686	782 0
33 0	283 0.74986	533 0.824847	783 0.339046
34 0	284 0.372235	534 0.202837	784 0.219045
35 0.287394	285 0	535 0.0472703	785 0.142218
36 0.644072	286 0.367373	536 0.286145	786 0
37 0.55737	287 0.471695	537 0.448244	787 0
38 0.308258	288 0.358583	538 0.0693457	788 0.172081
39 0.294114	289 0	539 0.840551	789 0.028594
40 0.078198	290 0.102332	540 0	790 0.925056
41 0.212997	291 0	541 0.0795548	791 0.216666
42 0.0854596	292 0	542 0	792 0
43 0.130041	293 0.133415	543 0.192297	793 0.109358
44 0.274695	294 0.28297	544 0	794 0.847948
45 0.21158	295 0	545 0	795 0.0361688
46 0	296 0	546 0.655395	796 0.314659
47 0	297 0.156736	547 0	797 0.183063
48 0.344381	298 0.0308611	548 0	798 0.228765
49 0	299 0.457095	549 0.848482	799 0.200963
50 0	300 0.248695	550 0	800 0
51 0.22524	301 0	551 0.740587	801 0.245673
52 0	302 0	552 0.259241	802 0
53 0	303 0	553 0.288728	803 0
54 0.342471	304 0.790683	554 0.379145	804 0
55 0	305 0	555 0.434649	805 0.290694
56 0.180932	306 0.340151	556 0.287973	806 0.340423
57 0.0882733	307 0	557 0.137395	807 0
58 0	308 0	558 0	808 0.0399604
59 0.375524	309 0.23116	559 0	809 0
60 0	310 0	560 0.380854	810 0.721054
61 0.351001	311 0	561 0.439529	811 0
62 0.231709	312 0	562 0.0598512	812 0.0866122
63 0.285059	313 0.0868251	563 0.0164353	813 0
64 0.40009	314 0.110039	564 0	814 0
65 0.12916	315 0.870975	565 0.445614	815 0.322403
66 0.33083	316 0.194509	566 0.504626	816 0.179806
67 0.295639	317 0.360498	567 0.00837891	817 0.368898
68 0.334848	318 0.838357	568 0	818 0
69 0	319 0.0392401	569 0.475034	819 0
70 0	320 0.274721	570 0	820 0.749432
71 0.0637517	321 0	571 0	821 0.201765
72 0.0416463	322 0	572 0	822 0.349142
73 0.447982	323 0.845298	573 0.0659888	823 0.629163
74 0	324 0	574 0.176023	824 0
75 0.167598	325 0	575 0.460648	825 0.74666
76 0.805018	326 0	576 0.780718	826 0
77 0.540831	327 0.214742	577 0	827 0
78 0.0311393	328 0	578 0	828 0.441648
79 0.298303	329 0	579 0	829 0
80 0	330 0	580 0.0453264	830 0
81 0	331 0	581 0.220442	831 0.159324
82 0.551498	332 0	582 0.189497	832 0.215277
83 0.512809	333 0.28534	583 0.117051	833 0
84 0.262398	334 0.066426	584 0.126489	834 0

85	0.0140733	335	0.229698	585	0	835	0.0320411
86	0	336	0.0362476	586	0.230969	836	0
87	0.399659	337	0.0708338	587	0	837	0.0447912
88	0	338	0	588	0	838	0
89	0.440365	339	0.319621	589	0.254582	839	0.214206
90	0.120963	340	0.532614	590	0	840	0.551333
91	0	341	0.232232	591	0.125605	841	0
92	0.134512	342	0	592	0.166988	842	0
93	0.109162	343	0.566005	593	0.75457	843	0
94	0	344	0.482786	594	0.00374597	844	0.626962
95	0.00889906	345	0.189914	595	0	845	0
96	0.0572528	346	0.294848	596	0.113495	846	0.0789927
97	0	347	0.235104	597	0.007323	847	0
98	0	348	0.640147	598	0	848	0.0596351
99	0	349	0.156822	599	0	849	0
100	0	350	0.232071	600	0.961795	850	0.111799
101	0.165129	351	0	601	0	851	0.0603805
102	0	352	0.520474	602	0	852	0
103	0.38556	353	0.286468	603	0	853	0.432873
104	0	354	0.1187	604	0	854	0
105	0	355	0.142851	605	0	855	0.206037
106	0	356	0.172139	606	0.0168101	856	0.0472856
107	0.480342	357	0	607	0	857	0.158492
108	0.32527	358	0.422909	608	0.341193	858	0
109	0	359	0	609	0	859	0.557548
110	0	360	0	610	0	860	0.134514
111	0	361	0.0905198	611	0.370238	861	0
112	0.188753	362	0.0702003	612	0	862	0.961126
113	0	363	0.0588756	613	0.232234	863	0
114	0.47325	364	0	614	0	864	0
115	0	365	0	615	0	865	0
116	0.400533	366	0.139395	616	0	866	0
117	0	367	0.344569	617	0.277526	867	0
118	0.16512	368	0	618	0	868	0.0756281
119	0.0644397	369	0	619	0.192505	869	0
120	0	370	0.183175	620	0.699973	870	0
121	0	371	0.0983092	621	0	871	0
122	0	372	0.173783	622	0.413688	872	0.188764
123	0.0419663	373	0.136237	623	0.281191	873	0
124	0.0138942	374	0.0815875	624	0	874	0.385433
125	0.634891	375	0	625	0.0587905	875	0
126	0.373056	376	0.0257261	626	0	876	0.759982
127	0.161799	377	0.319832	627	0.0129422	877	0
128	0.128518	378	0.310127	628	0.213121	878	0.88244
129	0	379	0.122282	629	0	879	0.0976339
130	0	380	0.147725	630	0.0976889	880	0.360442
131	0.542203	381	0.386029	631	0.167735	881	0.967868
132	0.251535	382	0	632	0.599425	882	0
133	0.108042	383	0.15892	633	0	883	0.0321553
134	0	384	0	634	0.0634269	884	0
135	0	385	0.335797	635	0	885	0
136	0.0693474	386	0.313676	636	0	886	0.0522611
137	0.273596	387	0.332336	637	0	887	0
138	0.136228	388	0.30649	638	0.676907	888	0
139	0	389	0.415413	639	0.0137484	889	0.70052

140 0	390 0.155605	640 0	890 0.193852
141 0	391 0.196735	641 0.492787	891 0.744535
142 0.634695	392 0	642 0.240658	892 0.128258
143 0.239357	393 0	643 0	893 0.413743
144 0.317891	394 0.590104	644 0.144862	894 0.0130363
145 0	395 0	645 0	895 0.205412
146 0	396 0.297974	646 0	896 0
147 0	397 0.256953	647 0	897 0
148 0.114831	398 0.0411328	648 0.414478	898 0.298437
149 0.110283	399 0.221826	649 0.423224	899 0.0834039
150 0.0909535	400 0	650 0.125797	900 0.354265
151 0.280392	401 0.135707	651 0	901 0.659423
152 0.30674	402 0.655971	652 0.338216	902 0.421396
153 0	403 0.051864	653 0	903 0.203881
154 0.185151	404 0	654 0	904 0
155 0.0549533	405 0	655 0	905 0
156 0.157467	406 0.162899	656 0.976392	906 0
157 0.241392	407 0.427351	657 0.920446	907 0.618651
158 0.200834	408 0.85125	658 0	908 0.156333
159 0.0667085	409 0.144152	659 0.192165	909 0
160 0	410 0.269275	660 0	910 0.18158
161 0.0511314	411 0.378316	661 0	911 0.122995
162 0.561656	412 0	662 0	912 0.431925
163 0.199165	413 0	663 0	913 0.390937
164 0.340151	414 0	664 0.282096	914 0
165 0	415 0	665 0	915 0.21761
166 0	416 0.0514366	666 0.0434762	916 0.0569363
167 0.281415	417 0.396392	667 0	917 0
168 0.171968	418 0	668 0	918 0.0430681
169 0.428484	419 0	669 0.298469	919 0
170 0	420 0	670 0	920 0.264985
171 0.39594	421 0.734734	671 0.208168	921 0
172 0	422 0	672 0.0217915	922 0.00578642
173 0.467471	423 0	673 0	923 0.328487
174 0.980161	424 0	674 0.597	924 0
175 0.232221	425 0.49511	675 0.147863	925 0.0119021
176 0.116414	426 0	676 0	926 0.174702
177 0.0154371	427 0.0290984	677 0	927 0
178 0.0811916	428 0.11759	678 0.528817	928 0
179 0	429 0	679 0	929 0.216071
180 0	430 0.4557	680 0	930 0.00879106
181 0.206413	431 0.308859	681 0.204401	931 0.111661
182 0.163111	432 0.331245	682 0	932 1.20092
183 0.360286	433 0	683 0	933 0
184 0	434 0.363471	684 0.572546	934 0
185 0	435 0.73493	685 0.418385	935 0.441016
186 0.02999	436 0.0702142	686 0	936 0
187 0.298306	437 0.363687	687 0.539342	937 0
188 0.509719	438 0.343231	688 0	938 0
189 0	439 0.146292	689 0.443328	939 0.0755847
190 0.475834	440 0.0295736	690 0.100322	940 0.160823
191 0	441 0.367307	691 0	941 0
192 0	442 0	692 0	942 0
193 0.119643	443 0.412691	693 0.732499	943 0
194 0	444 0	694 0.22996	944 0.499266

195 0	445 0.327447	695 0.894006	945 0
196 0	446 0.425299	696 0	946 0
197 0.518999	447 0.0912591	697 0	947 0
198 0	448 0.195418	698 0	948 0
199 0	449 0	699 0.13648	949 0.501789
200 0.572819	450 0	700 0	950 0.080078
201 0.186225	451 0	701 0	951 0
202 0	452 0.582448	702 0.211928	952 0.177317
203 0.359171	453 0.132159	703 0	953 0
204 0.569268	454 0.286281	704 0.276625	954 0
205 0.258718	455 0.069628	705 0	955 0
206 0	456 0	706 0.186289	956 0.0343649
207 0.154279	457 0.279299	707 0.0453639	957 0.35326
208 0	458 0.0246735	708 0.0923635	958 0
209 0.0548216	459 0.377721	709 0.377031	959 0
210 0	460 0	710 0.057472	960 0.623813
211 0	461 0	711 0.168894	961 0.265106
212 0	462 0.303348	712 0.231532	962 0
213 0	463 0.70188	713 0.258557	963 0
214 0.146342	464 0.167952	714 0	964 0.503417
215 0	465 0	715 0.336567	965 0
216 0.375863	466 0	716 0	966 0
217 0	467 0.0145906	717 0.134234	967 0.0970009
218 0	468 0	718 0	968 0
219 0.345899	469 0	719 0.0293412	969 0.578538
220 0	470 0	720 1.12497	970 0.378364
221 0	471 0	721 0	971 0.478624
222 0	472 0.638816	722 0.0537692	972 0
223 0.0801721	473 0	723 0	973 0.136918
224 0.189696	474 0	724 0	974 0.45531
225 0.313312	475 0	725 0	975 0.350487
226 0.153588	476 0.158179	726 0.0147402	976 0.284055
227 0.652083	477 0.0962412	727 0	977 0.365105
228 0	478 0.189305	728 0	978 0.3715
229 0.264539	479 0	729 1.1739	979 0
230 0.482231	480 0.596667	730 0	980 0.196374
231 0.15863	481 0.450151	731 0	981 0.305205
232 0.339995	482 0	732 0	982 0.0325256
233 0.720511	483 0.240785	733 0	983 0.230625
234 0	484 0.247176	734 0.397895	984 0
235 0	485 0.322611	735 0	985 0
236 0	486 0.28826	736 0.0727223	986 0.380838
237 0.0258435	487 0.497746	737 0.258928	987 0.828729
238 0.387668	488 0.39768	738 0.793016	988 0
239 0	489 0.158332	739 0	989 0.295836
240 0.0333182	490 0	740 0.180385	990 0.254667
241 0	491 0.0786052	741 0	991 0.272515
242 0.347115	492 0.393665	742 0.396433	992 0.751416
243 0.588083	493 0	743 0.376752	993 0.205435
244 0.0538069	494 0	744 0	994 0
245 0	495 0.395627	745 0	995 0.163205
246 0.00697644	496 0.0807348	746 0	996 0.236692
247 0	497 0.0474954	747 0.417673	997 0
248 0	498 0.22097	748 0.300632	998 0.50451
249 0.195517	499 0.173559	749 0	999 0.426533

```
 250 0             500 0.654088      750 0             1000 0.15504
;
```

ampl:

The SVM output was tested using MATLAB with the code below:

Test3.m

```
clear all; clc;
ans = dlmread('Train_Set_Matlab.txt');
Train_Set = ans(:,3:end);
ans = dlmread('Test_Set_Matlab.txt');
Test_Set = ans(:,3:end);

op = dlmread('output.txt');
sv = [];
for (i = 1:length(op))
if(op(i,2)>0)
sv = [sv;op(i,1)];
end
end
sv_alpha = op(sv,2);
sv_y = op(sv,3);
support_vectors = [sv,sv_alpha,sv_y];

%Forming the kernel
Kernel = zeros(length(Train_Set));
gama = 0.05;
for(i = 1:length(Train_Set))
    xi = Train_Set(i,:);
    for (j = 1:length(Train_Set))
        xj = Train_Set(j,:);
        k = exp(-gama*norm(xi-xj,2)^2);
        Kernel(i,j) = k;
    end
end


%Calculating b for each of the vectors to check if all are equal
for(i = 1:length(support_vectors))
    summation = 0;
    for(j = 1:length(Train_Set))
        summation = summation+op(j,2)*op(j,3)*Kernel(j,support_vectors(i,1));
    end
    b = summation - support_vectors(i,3);
    B(i) = b;
end


%Testing if alpha obtained is correct
for(i = 1:length(Test_Set))
    sum = 0;
    for (j = 1:length(support_vectors))
```

```
        sum = sum +
(support_vectors(j,3)*support_vectors(j,2)*(Train_Set(support_vectors(j,1),:)
*Test_Set(i,:)'));
    end
    sum = sum - b;
    if(sum >= 0)
        y_result(i) = 1;
    else
        y_result(i)=-1;
    end
end
y_result=y_result';

hits = 0;
misses = 0;
for (i = 1:length(y_result))
    if(y_result(i) == op(i,3))
        hits = hits+1;
    else
        misses = misses+1;
    end
end
display ('The Value of b for various support vectors:');
B

display ('Checking the Obtained values of Y from Test data to verify accuracy
and errors')
hits
misses

Accuracy_Percent = hits/10
Error_Percent = misses/10
```

The Test output was as follows:

Test4.m : Output for Testing SVM-4.mod

```
The Value of b for various support vectors:


B =


  Columns 1 through 10


   -0.1573   -0.1573   -0.1573   -0.1573   -0.1573   -0.1573   -0.1573   -0.1573   -
0.1573   -0.1573


  Columns 11 through 20
```

-0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573

Columns 21 through 30

-0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573

Columns 31 through 40

-0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573

Columns 41 through 50

-0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573

Columns 51 through 60

-0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573

… … …

Columns 571 through 580

-0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573 -0.1573

Column 581

-0.1573

```
Checking the Obtained values of Y from Test data to verify accuracy and errors

hits =

   976

misses =

    24


Accuracy_Percent =

   97.6000

Error_Percent =

    2.4000


>>
```

Dual-soft Margin SVM with Radial Basis Kernel to differentiate Even from Odd

The training code for this question was implemented in AMPL as follows:

SVM-5.mod

```
reset;
param n := 785;
param D := 1000;
param C := 1000;
param gama := 0.05;

set POINTS := {2..n};
set INPUT_POINTS := {1..n};
set DATASET := {1..D};

param x{DATASET,INPUT_POINTS};
param y{DATASET};

data Train_Set_Full.txt;

#Initializing KERNEL Function: Radial-basis function machine with K[i,j] = e^(-
g*(x[j]-x[i])^2)
param K{i in DATASET, j in DATASET} := exp (-gama*(sum{t in POINTS} (x[i,t]-
x[j,t])^2));

#Initializing Y[i]
for {i in DATASET} {
```

```
        #Determining if the digit is ODD(=1) or EVEN(=-1)
        if ((x[i,1] mod 2) = 1) then {let y[i] := 1} else{ let y[i] := -1};

}

var alpha{DATASET} >= 0, <= C;   #

maximize DUALSVM : (sum {i in DATASET} alpha[i]) - 0.5*(sum {i in DATASET,j in
DATASET} alpha[i]*alpha[j]*y[i]*y[j]*K[i,j]);

s.t. CONDITION :  sum {i in DATASET} alpha[i]*y[i] = 0;

option solver snopt;
solve;
```

The values of support vectors obtained were as follows:

SVM-5.mod: Output

```
ampl: model SVM-5.mod;
SNOPT 7.2-8 : Optimal solution found.
561 iterations, objective 24727290.29
alpha [*] :
   1       0           335       0           669   88742.9
   2       1.08293     336       0           670       0
   3      12.4707      337       0           671       0
   4      12.4005      338       0           672       0
   5       0           339       0           673       0
   6      11.1521      340       0           674       0
   7       0           341       0           675       0
   8       0           342  241127           676       0
   9       2.89783     343       0           677       0
  10      10.63        344       0           678       0
  11       0           345       0           679       0
  12      15.3639      346       0           680       0
  13       0           347       0           681       0
  14       0           348       0           682       0
  15       0           349       0           683       0
  16       0           350       0           684       0
  17       0           351       0           685       0
  18       0.122802    352  849474           686       0
  19      11.3511      353       0           687       0
  20       0           354       0           688       0
  21       2.87149     355       0           689       0
  22       0           356       0           690       0
  23       0           357       0           691       0
  24       3.19399     358       0           692       0
  25       3.49242     359  506467           693       0
  26       7.68065     360       0           694       0
  27       9.23025     361       0           695       0
  28       4.85517     362       0           696       0
```

29	7.93756	363	0	697	0
30	12.2203	364	0	698	0
31	3.62271	365	0	699	0
32	0	366	0	700	0
33	4.45621	367	0	701	0
34	0	368	0	702	0
35	9.52635	369	0	703	0
36	5.50384	370	0	704	0
37	0	371	0	705	0
38	2.62178	372	86252	706	0
39	0	373	0	707	0
40	0	374	0	708	0
41	0	375	0	709	0
42	0	376	0	710	3289.84
43	0	377	0	711	7072.44
44	1.28197	378	0	712	0
45	0	379	0	713	0
46	11.3503	380	113591	714	0
47	7.10089	381	742664	715	0
48	9.47784	382	0	716	0
49	4.83504	383	828412	717	0
50	5.01973	384	0	718	0
51	0	385	1513.04	719	14460.2
52	8.0308	386	0	720	70178.7
53	0	387	0	721	23319.3
54	0	388	0	722	0
55	0	389	1546.87	723	0
56	5.35372	390	446411	724	0
57	0	391	0	725	0
58	0	392	0	726	0
59	0	393	0	727	0
60	0	394	0	728	0
61	0	395	0	729	0
62	0	396	0	730	13100.3
63	0	397	320425	731	46817.9
64	3.46969	398	0	732	0
65	0	399	0	733	0
66	10.5403	400	0	734	50398
67	0	401	25347.9	735	0
68	0	402	929760	736	0
69	0	403	0	737	0
70	0	404	2719010	738	0
71	0	405	0	739	12744.6
72	0	406	63616.7	740	0
73	5.98871	407	0	741	0
74	1.69913	408	245296	742	0
75	13.4808	409	0	743	0
76	5.52811	410	0	744	0
77	5.29266	411	0	745	0
78	2.70554	412	0	746	0
79	16.5686	413	0	747	0
80	15.0955	414	0	748	0
81	2.94544	415	0	749	0
82	8.62111	416	0	750	0
83	11.0779	417	767566	751	27623.1

84	11.2127	418	0	752	0
85	13.4749	419	0	753	0
86	0	420	0	754	18158.7
87	9.95221	421	0	755	0
88	4.49409	422	360497	756	0
89	8.59351	423	388907	757	0
90	3.33396	424	0	758	0
91	0	425	0	759	0
92	0	426	454085	760	0
93	9.7331	427	0	761	145935
94	8.61736	428	0	762	0
95	0	429	0	763	0
96	0	430	1140240	764	0
97	0	431	424519	765	0
98	0	432	0	766	0
99	0	433	0	767	0
100	0	434	0	768	0
101	0	435	0	769	69627.8
102	0	436	0	770	0
103	0	437	2216500	771	0
104	48.4622	438	0	772	0
105	0	439	923847	773	0
106	0	440	0	774	0
107	60.3446	441	0	775	1080.76
108	0	442	412533	776	0
109	0	443	0	777	0
110	0	444	288965	778	0
111	13.1709	445	0	779	80628.2
112	0	446	943522	780	0
113	0	447	1211220	781	0
114	0	448	0	782	19186.8
115	0	449	873599	783	0
116	0	450	0	784	0
117	0	451	660128	785	0
118	11.8408	452	1905280	786	0
119	0	453	0	787	0
120	0	454	0	788	0
121	0	455	0	789	0
122	0	456	0	790	0
123	0	457	0	791	0
124	128.046	458	660918	792	0
125	0	459	0	793	0
126	3.58183	460	0	794	0
127	17.083	461	0	795	0
128	0	462	0	796	5102.45
129	0	463	1056710	797	0
130	22.5777	464	0	798	0
131	0	465	0	799	23135.1
132	0	466	0	800	0
133	149.27	467	0	801	0
134	0	468	252660	802	0
135	65.7512	469	144278	803	0
136	0	470	429030	804	0
137	0	471	0	805	0
138	27.7326	472	0	806	1980.62

139	0	473	92188.4	807	0
140	40.0632	474	0	808	1535.1
141	68.1525	475	0	809	0
142	0	476	0	810	302.788
143	0	477	0	811	2140.55
144	12.637	478	102762	812	0
145	92.7699	479	0	813	0
146	0.547909	480	0	814	5145.71
147	0	481	0	815	0
148	0	482	132123	816	0
149	18.3213	483	426436	817	0
150	0	484	0	818	813.255
151	0	485	0	819	0
152	10.1877	486	765734	820	9050.28
153	89.5913	487	0	821	4494.19
154	0	488	550499	822	0
155	0	489	0	823	0
156	0	490	0	824	0
157	0	491	0	825	0
158	0	492	0	826	0
159	0	493	0	827	0
160	0	494	0	828	0
161	0	495	0	829	3613.49
162	0	496	17609.2	830	0
163	0	497	234910	831	0
164	17.6628	498	0	832	0
165	0	499	0	833	0
166	18.2793	500	0	834	0
167	0	501	0	835	7543.87
168	0	502	21957	836	0
169	0	503	0	837	0
170	0	504	831141	838	5334.28
171	0	505	0	839	3396.68
172	0	506	0	840	0
173	90.5567	507	0	841	261.669
174	11.8555	508	520598	842	48.0613
175	45.9337	509	0	843	0
176	74.2487	510	0	844	0
177	0	511	53181.9	845	0
178	111.89	512	0	846	0
179	0	513	0	847	0
180	97.8517	514	0	848	0
181	0	515	0	849	1616.49
182	0	516	0	850	0
183	291.636	517	395556	851	0
184	0	518	0	852	0
185	0	519	0	853	0
186	0	520	0	854	0
187	287.309	521	0	855	0
188	0	522	0	856	0
189	202.8	523	0	857	0
190	0	524	0	858	0
191	48.5509	525	0	859	883.665
192	0	526	1136210	860	1913.46
193	0	527	0	861	0

194	0	528	560640	862	0
195	0	529	0	863	0
196	0	530	0	864	0
197	0	531	0	865	0
198	0	532	264583	866	0
199	0	533	0	867	1028.18
200	0	534	0	868	0
201	88000.6	535	0	869	0
202	33453.9	536	382690	870	0
203	0	537	0	871	0
204	0	538	474714	872	0
205	192591	539	0	873	0
206	0	540	0	874	0
207	0	541	840365	875	0
208	0	542	0	876	0
209	0	543	0	877	0
210	0	544	0	878	0
211	182490	545	1522070	879	0
212	0	546	0	880	0
213	0	547	0	881	0
214	0	548	0	882	5015.49
215	187803	549	0	883	0
216	0	550	0	884	2825.64
217	0	551	77074	885	0
218	78823.8	552	0	886	0
219	0	553	586503	887	0
220	0	554	0	888	0
221	0	555	0	889	1463.59
222	0	556	0	890	0
223	0	557	229362	891	0
224	0	558	990283	892	0
225	0	559	0	893	0
226	0	560	252641	894	0
227	0	561	0	895	0
228	108941	562	0	896	0
229	0	563	0	897	467.28
230	0	564	0	898	0
231	0	565	0	899	0
232	0	566	0	900	2326.96
233	0	567	0	901	0
234	0	568	0	902	22.1341
235	0	569	1507420	903	135.602
236	0	570	773561	904	144.881
237	0	571	0	905	82.3009
238	0	572	676212	906	0.930635
239	0	573	0	907	0
240	0	574	0	908	10.875
241	20478.5	575	38404.8	909	2.11469
242	0	576	0	910	0
243	0	577	0	911	0
244	0	578	797614	912	0
245	94811.9	579	0	913	0
246	0	580	0	914	0
247	0	581	118456	915	63.0542
248	0	582	0	916	0

249	0	583	0	917	17.855
250	0	584	0	918	0
251	0	585	0	919	0
252	0	586	0	920	0
253	0	587	0	921	87.6325
254	0	588	912701	922	17.3897
255	0	589	0	923	54.7555
256	0	590	1208000	924	0
257	0	591	0	925	124.724
258	0	592	0	926	0
259	0	593	0	927	0
260	0	594	802802	928	29.0404
261	0	595	0	929	0
262	88427.4	596	0	930	18.4927
263	0	597	0	931	13.155
264	0	598	0	932	0
265	15742.3	599	0	933	0
266	0	600	0	934	0
267	127388	601	0	935	25.0522
268	0	602	28085.1	936	3.14903
269	0	603	0	937	0.892668
270	0	604	0	938	91.2667
271	99858.6	605	157184	939	0
272	0	606	207718	940	0
273	0	607	0	941	0
274	0	608	0	942	0
275	40235.8	609	0	943	0
276	0	610	0	944	28.2235
277	0	611	0	945	0
278	0	612	0	946	0
279	117616	613	0	947	0
280	0	614	0	948	197.072
281	105510	615	0	949	63.4595
282	69946.1	616	231902	950	0
283	0	617	0	951	0
284	0	618	0	952	0
285	0	619	0	953	0
286	0	620	0	954	0
287	0	621	0	955	0
288	0	622	0	956	195.092
289	0	623	0	957	0
290	0	624	0	958	0
291	0	625	0	959	37.6753
292	0	626	0	960	0
293	0	627	0	961	14.6176
294	0	628	0	962	0
295	0	629	0	963	0
296	0	630	0	964	0
297	3161.3	631	0	965	0
298	0	632	0	966	16.3822
299	36346.6	633	0	967	67.4438
300	0	634	0	968	0
301	0	635	0	969	0
302	0	636	0	970	0
303	0	637	0	971	0

```
 304       0          638       0          972       0
 305       0          639       0          973      27.5051
 306       0          640       0          974       0
 307       0          641       0          975       0
 308       0          642       0          976      22.5344
 309       0          643       0          977       0
 310  168848          644       0          978       0
 311       0          645       0          979      91.329
 312       0          646   79434.7        980       2.93459
 313  680078          647  101294          981       0
 314       0          648       0          982      45.4414
 315  874624          649       0          983       0
 316  668101          650       0          984       0
 317       0          651       0          985       0
 318       0          652       0          986      24.462
 319  223951          653       0          987       0
 320       0          654       0          988      90.6672
 321       0          655       0          989       0
 322       0          656       0          990      26.871
 323       0          657       0          991       0
 324       0          658       0          992       1.39968
 325       0          659  194386          993      25.3075
 326  649942          660       0          994      37.3718
 327       0          661       0          995       0
 328       0          662       0          996       0
 329       0          663   16342.6        997       5.12634
 330  589137          664       0          998      51.089
 331   87696.6        665       0          999       0
 332       0          666   34641.1       1000       0
 333       0          667   12056.7
 334   36235.2        668       0
;
ampl:
```

The code used for Testing was as follows:

Test5.m

```
clear all; clc;
Train_Set = dlmread('Train_Set_Matlab.txt');
Train_Set = Train_Set(2:end,2:end);
Test_Set = dlmread('Test_Set_Full.txt');
Test_Set = Test_Set(2:end,2:end);

op = dlmread('output.txt');
sv = [];
%nop = [op(:,1:2);op(:,3:4);op(:,5:6);op(:,7:8)];
%op = nop;
for (i = 1:length(op))
if(op(i,2)>0 && op(i,2) < 100)
sv = [sv;op(i,1)];
end
end
sv_alpha = op(sv,2);
sv_y = op(sv,3);
```

```
support_vectors = [sv,sv_alpha,sv_y];

%Forming the kernel
Kernel = zeros(length(Train_Set)-1);
gama = 0.05;
for(i = 1:length(Train_Set))
    xi = Train_Set(i,:);
    for (j = 1:length(Train_Set))
        xj = Train_Set(j,:);
        k = exp(-gama*norm(xi-xj,2)^2);
        Kernel(i,j) = k;
    end
end


%Calculating b for each of the vectors to check if all are equal
for(i = 1:length(support_vectors))
    summation = 0;
    for(j = 1:length(Train_Set))
        summation = summation+op(j,2)*op(j,3)*Kernel(j,support_vectors(i,1));
    end
    b = summation - support_vectors(i,3);
    B(i) = b;
end


%Testing if alpha obtained is correct
for(i = 1:length(Test_Set))
    sum = 0;
    for (j = 1:length(support_vectors))
        sum = sum +
(support_vectors(j,3)*support_vectors(j,2)*(Train_Set(support_vectors(j,1),:)
*Test_Set(i,:)'));
    end
    sum = sum - b;
    if(sum >= 0)
        y_result(i) = 1;
    else
        y_result(i)=-1;
    end
end
y_result=y_result';
misses = 0;
hits = 0;
for (i = 1:length(y_result))
    if(y_result(i)-op(i,3) ~= 0)
        misses = misses+1;
    else
        hits = hits+1;
    end
end
B
fprintf ('Accuracy Percent: %f \n', hits/10);
fprintf('Error Percent: %f \n', misses/10);
```

The Testing Results were as follows:

SVM-5: Radial Basis, Even/Odd – Values of B and Test Accuracy

B =

Columns 1 through 10

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 11 through 20

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 21 through 30

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 31 through 40

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 41 through 50

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 51 through 60

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

... ...

Columns 801 through 810

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 811 through 820

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 821 through 830

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 831 through 840

-0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650 -0.2650

Columns 841 through 844

-0.2650 -0.2650 -0.2650 -0.2650

Accuracy Percent: 81.000000

Error Percent: 19.000000