SUPPORT VECTOR MACHINES

To investigate SVMs we'll use a dataset regarding customer churn from a local cell phone company. The company has provided the following variables:

- U.S. State
- Length of account in days
- Phone Area Code (3 levels)
- International Plan (Binary)
- Voicemail Plan (Binary)
- Number of voicemail messages
- Total day minutes
- Total day calls
- Total daytime charges
- Total evening minutes
- Total evening charges
- Total evening calls
- Total nighttime minutes
- Total nighttime charges
- Total nighttime calls
- Total international minutes
- Total international charges
- Total international calls
- Number of customer service calls
- Churn - (a binary target)

We can try to predict customer churn using any of the classification methods discussed previously in this class. Since this is a relatively small training set (3333 observations), it will be practical to model it using SVMs and Kernals.

Tuning a Support Vector Machine with RBF Kernal and Regularization

The e1071 package comes with a nice function to perform a grid search over the parameters γ and C. All we need to do is input our centered and scaled data, our model, and a vector of each parameter for which the method will try every combination and report the average performance on cross-validation. The following code could take between 1 and 3 minutes to run on an average computer. Note: there are many different approaches to scaling data that could be beneficial. Sometimes a simple linear scaling of the features into a range of [-1,1] works better than statistical standardization. Here we use statistical standardization on the interval variables and leave the categorical variables alone.

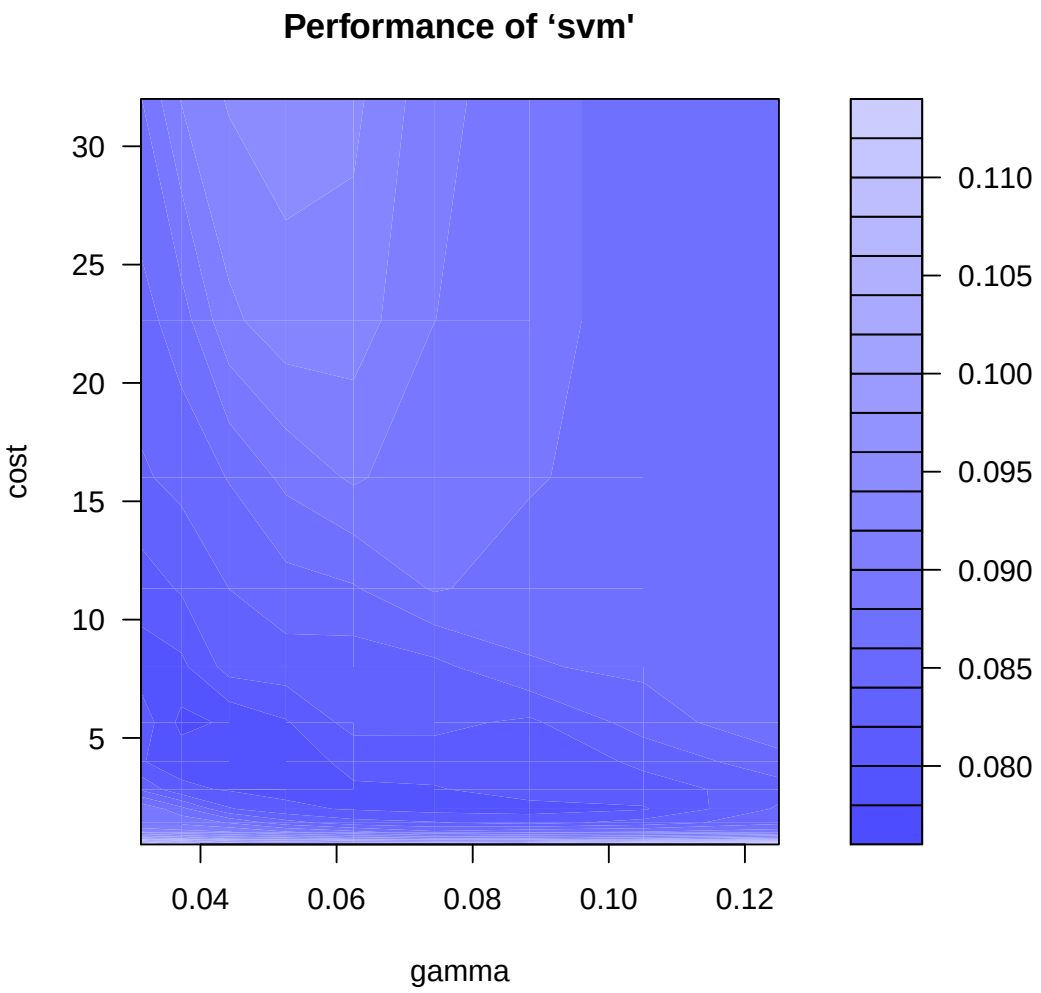
```
> load('churn.Rdata')
> churnTrainScale = scale(churnTrain[,c(2,6:18)], center=T, scale=T)
> churnTrainScale = cbind(churnTrain[,c(1,3:5,19,20)],churnTrainScale)
> churnTestScale = scale(churnTest[,c(2,6:18)], center=T, scale=T)
> churnTestScale = cbind(churnTest[,c(1,3:5,19,20)],churnTestScale)
> library(e1071)
> TuneSVM = tune.svm(churn~., data=churnTrainScale,kernel='radial', gamma=2^seq(-5,-3,0.25), cost=2^seq(-1,5,0.5))
> summary(TuneSVM)
```

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
      gamma     cost
 0.03716272 5.656854

- best performance: 0.07740825

- Detailed performance results:
         gamma       cost      error  dispersion
1   0.03125000  0.5000000 0.11071281 0.020405322
2   0.03716272  0.5000000 0.10621010 0.020957428
3   0.04419417  0.5000000 0.10410890 0.020991075
4   0.05255603  0.5000000 0.10260920 0.022139453
5   0.06250000  0.5000000 0.10291040 0.021518601
6   0.07432544  0.5000000 0.10261189 0.021742411
7   0.08838835  0.5000000 0.10531190 0.020685536
8   0.10511205  0.5000000 0.10951251 0.022078914
9   0.12500000  0.5000000 0.11371222 0.021904966
10  0.03125000  0.7071068 0.10230800 0.021152918
11  0.03716272  0.7071068 0.10110859 0.019519385
12  0.04419417  0.7071068 0.09870979 0.018004720
13  0.05255603  0.7071068 0.09811009 0.017872071
14  0.06250000  0.7071068 0.09661218 0.016656446
15  0.07432544  0.7071068 0.09571308 0.015904428
16  0.08838835  0.7071068 0.09511338 0.015878991
17  0.10511205  0.7071068 0.09691338 0.013649383
18  0.12500000  0.7071068 0.10021399 0.015381316
19  0.03125000  1.0000000 0.09451008 0.012016435
20  0.03716272  1.0000000 0.09301397 0.012989438
```

21	0.04419417	1.0000000	0.09241337	0.013791854
22	0.05255603	1.0000000	0.09061067	0.013329142
23	0.06250000	1.0000000	0.08911187	0.012972527
24	0.07432544	1.0000000	0.08671306	0.010275188
25	0.08838835	1.0000000	0.08821276	0.013889947
26	0.10511205	1.0000000	0.08851037	0.015176953
27	0.12500000	1.0000000	0.08941037	0.013984972
28	0.03125000	1.4142136	0.08911366	0.011954945
29	0.03716272	1.4142136	0.08941307	0.010310532
30	0.04419417	1.4142136	0.08791336	0.009424051
31	0.05255603	1.4142136	0.08490946	0.008728732
32	0.06250000	1.4142136	0.08280916	0.009514760
33	0.07432544	1.4142136	0.08221215	0.011536097
34	0.08838835	1.4142136	0.08131485	0.011660126
35	0.10511205	1.4142136	0.08281545	0.013397604
36	0.12500000	1.4142136	0.08551186	0.011448053
37	0.03125000	2.0000000	0.08971337	0.012939142
38	0.03716272	2.0000000	0.08671306	0.012056039
39	0.04419417	2.0000000	0.08221215	0.014371318
40	0.05255603	2.0000000	0.08041215	0.013188240
41	0.06250000	2.0000000	0.07981245	0.012606896
42	0.07432544	2.0000000	0.07921185	0.010253929
43	0.08838835	2.0000000	0.07921364	0.012136971
44	0.10511205	2.0000000	0.07981514	0.011893017
45	0.12500000	2.0000000	0.08431425	0.011825323
46	0.03125000	2.8284271	0.08371096	0.013383660
47	0.03716272	2.8284271	0.08041215	0.012413433
48	0.04419417	2.8284271	0.07981155	0.011434843
49	0.05255603	2.8284271	0.07951215	0.012691136
50	0.06250000	2.8284271	0.07981335	0.011460164
51	0.07432544	2.8284271	0.07981425	0.010169237
52	0.08838835	2.8284271	0.08101365	0.010529981
53	0.10511205	2.8284271	0.08101455	0.012917923
54	0.12500000	2.8284271	0.08311485	0.012919788
55	0.03125000	4.0000000	0.08010915	0.012408404
56	0.03716272	4.0000000	0.07921095	0.011605362
57	0.04419417	4.0000000	0.07981155	0.012361218
58	0.05255603	4.0000000	0.07921185	0.011266034
59	0.06250000	4.0000000	0.08041575	0.011439073
60	0.07432544	4.0000000	0.08101455	0.010998741
61	0.08838835	4.0000000	0.08011634	0.013465370
62	0.10511205	4.0000000	0.08251695	0.012718975
63	0.12500000	4.0000000	0.08521426	0.013153520
64	0.03125000	5.6568542	0.08130765	0.011339056
65	0.03716272	5.6568542	0.07740825	0.011641918
66	0.04419417	5.6568542	0.07831364	0.011743107
67	0.05255603	5.6568542	0.07981425	0.012463939
68	0.06250000	5.6568542	0.08281365	0.013155221
69	0.07432544	5.6568542	0.08251425	0.011888406
70	0.08838835	5.6568542	0.08161964	0.014333773
71	0.10511205	5.6568542	0.08491935	0.013031570
72	0.12500000	5.6568542	0.08761576	0.014644536
73	0.03125000	8.0000000	0.07890705	0.013556362
74	0.03716272	8.0000000	0.07951035	0.012526745
75	0.04419417	8.0000000	0.08281095	0.012190303
76	0.05255603	8.0000000	0.08311485	0.013819153
77	0.06250000	8.0000000	0.08281455	0.011547617
78	0.07432544	8.0000000	0.08341785	0.012211538
79	0.08838835	8.0000000	0.08581935	0.013719173

```
80  0.10511205  8.0000000 0.08642085 0.013553143
81  0.12500000  8.0000000 0.08791516 0.013743268
82  0.03125000 11.3137085 0.08100915 0.013425438
83  0.03716272 11.3137085 0.08221125 0.012112665
84  0.04419417 11.3137085 0.08401036 0.012651935
85  0.05255603 11.3137085 0.08521246 0.009435020
86  0.06250000 11.3137085 0.08581396 0.011102750
87  0.07432544 11.3137085 0.08821906 0.012256704
88  0.08838835 11.3137085 0.08701965 0.013849943
89  0.10511205 11.3137085 0.08731995 0.013446768
90  0.12500000 11.3137085 0.08731456 0.013400067
91  0.03125000 16.0000000 0.08370916 0.011445579
92  0.03716272 16.0000000 0.08461006 0.011901950
93  0.04419417 16.0000000 0.08611156 0.011849153
94  0.05255603 16.0000000 0.08851576 0.010378556
95  0.06250000 16.0000000 0.09032026 0.012688344
96  0.07432544 16.0000000 0.08852205 0.014241329
97  0.08838835 16.0000000 0.08821996 0.012970596
98  0.10511205 16.0000000 0.08702055 0.013564530
99  0.12500000 16.0000000 0.08731456 0.013400067
100 0.03125000 22.6274170 0.08520886 0.011926404
101 0.03716272 22.6274170 0.08701246 0.011683472
102 0.04419417 22.6274170 0.09151517 0.008694690
103 0.05255603 22.6274170 0.09331787 0.010223871
104 0.06250000 22.6274170 0.09302386 0.014717389
105 0.07432544 22.6274170 0.09002356 0.014506297
106 0.08838835 22.6274170 0.08882146 0.014015699
107 0.10511205 22.6274170 0.08702055 0.013564530
108 0.12500000 22.6274170 0.08731456 0.013400067
109 0.03125000 32.0000000 0.08791156 0.013207031
110 0.03716272 32.0000000 0.09211397 0.008428276
111 0.04419417 32.0000000 0.09421967 0.009184923
112 0.05255603 32.0000000 0.09482297 0.013227961
113 0.06250000 32.0000000 0.09452446 0.015009357
114 0.07432544 32.0000000 0.09062326 0.014491245
115 0.08838835 32.0000000 0.08882146 0.014015699
116 0.10511205 32.0000000 0.08702055 0.013564530
117 0.12500000 32.0000000 0.08731456 0.013400067
```

```
> plot(TuneSVM)
```



Then we can create the SVM model using the chosen parameters and test it on the validation data:

```
> bestgamma=0.0625
> bestc=4
> svm1=svm(churn~., data=churnTrainScale,type='C', kernel='radial', gamma=bestgamma, cost=bestc)
> pred=predict(svm1,churnTestScale)
> sum(pred!=churnTestScale$churn)/1667
```

```
[1] 0.06838632
```