svm

```
library(tidyverse)
```

```
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(stringr)
library(e1071)
```

```
heart <- read.csv(file = 'heart.csv')
heart %>%
  count(Cholesterol)
```

```
##     Cholesterol   n
## 1             0 172
## 2            85   1
## 3           100   2
## 4           110   1
## 5           113   1
## 6           117   1
## 7           123   1
## 8           126   2
## 9           129   1
## 10          131   1
## 11          132   1
## 12          139   2
## 13          141   1
## 14          142   1
## 15          147   2
## 16          149   2
## 17          152   1
## 18          153   1
## 19          156   1
## 20          157   1
## 21          159   1
## 22          160   6
## 23          161   2
## 24          163   2
## 25          164   2
```

```
## 26          165   1
## 27          166   4
## 28          167   3
## 29          168   2
## 30          169   2
## 31          170   2
## 32          171   3
## 33          172   2
## 34          173   2
## 35          174   1
## 36          175   4
## 37          176   1
## 38          177   6
## 39          178   1
## 40          179   2
## 41          180   3
## 42          181   2
## 43          182   5
## 44          183   1
## 45          184   4
## 46          185   3
## 47          186   6
## 48          187   2
## 49          188   4
## 50          190   2
## 51          192   4
## 52          193   6
## 53          194   2
## 54          195   7
## 55          196   6
## 56          197   7
## 57          198   6
## 58          199   3
## 59          200   4
## 60          201   6
## 61          202   3
## 62          203   7
## 63          204   9
## 64          205   3
## 65          206   3
## 66          207   6
## 67          208   7
## 68          209   5
## 69          210   4
## 70          211   9
## 71          212   6
## 72          213   7
## 73          214   7
## 74          215   6
## 75          216   9
## 76          217   4
## 77          218   6
## 78          219   8
## 79          220  10
```

```
## 80          221   5
## 81          222   6
## 82          223  10
## 83          224   6
## 84          225   7
## 85          226   6
## 86          227   4
## 87          228   5
## 88          229   4
## 89          230   9
## 90          231   5
## 91          232   3
## 92          233   6
## 93          234   7
## 94          235   5
## 95          236   6
## 96          237   6
## 97          238   4
## 98          239   4
## 99          240   8
## 100         241   4
## 101         242   2
## 102         243   7
## 103         244   4
## 104         245   6
## 105         246   8
## 106         247   3
## 107         248   6
## 108         249   5
## 109         250   5
## 110         251   1
## 111         252   3
## 112         253   4
## 113         254  11
## 114         255   3
## 115         256   5
## 116         257   3
## 117         258   7
## 118         259   2
## 119         260   8
## 120         261   3
## 121         262   1
## 122         263   8
## 123         264   6
## 124         265   4
## 125         266   4
## 126         267   5
## 127         268   5
## 128         269   6
## 129         270   6
## 130         271   4
## 131         272   3
## 132         273   5
## 133         274   6
```

```
## 134          275   7
## 135          276   4
## 136          277   5
## 137          278   1
## 138          279   1
## 139          280   2
## 140          281   3
## 141          282   7
## 142          283   5
## 143          284   4
## 144          285   2
## 145          286   2
## 146          287   2
## 147          288   6
## 148          289   6
## 149          290   2
## 150          291   3
## 151          292   4
## 152          293   1
## 153          294   4
## 154          295   5
## 155          297   4
## 156          298   5
## 157          299   2
## 158          300   2
## 159          302   2
## 160          303   4
## 161          304   2
## 162          305   4
## 163          306   3
## 164          307   2
## 165          308   6
## 166          309   4
## 167          310   3
## 168          311   2
## 169          312   2
## 170          313   1
## 171          315   3
## 172          316   1
## 173          318   3
## 174          319   1
## 175          320   2
## 176          321   1
## 177          322   1
## 178          325   2
## 179          326   2
## 180          327   1
## 181          328   1
## 182          329   1
## 183          330   2
## 184          331   1
## 185          333   1
## 186          335   2
## 187          336   1
```

```
## 188         337   1
## 189         338   1
## 190         339   2
## 191         340   2
## 192         341   3
## 193         342   3
## 194         344   1
## 195         347   1
## 196         349   1
## 197         353   1
## 198         354   1
## 199         355   1
## 200         358   1
## 201         360   1
## 202         365   1
## 203         369   1
## 204         384   1
## 205         385   1
## 206         388   1
## 207         392   1
## 208         393   1
## 209         394   2
## 210         404   1
## 211         407   1
## 212         409   1
## 213         412   1
## 214         417   1
## 215         458   1
## 216         466   1
## 217         468   1
## 218         491   1
## 219         518   1
## 220         529   1
## 221         564   1
## 222         603   1
```

```
heart %>%
  count(RestingBP)
```

```
##    RestingBP   n
## 1          0   1
## 2         80   1
## 3         92   1
## 4         94   2
## 5         95   6
## 6         96   1
## 7         98   1
## 8        100  15
## 9        101   1
## 10       102   3
## 11       104   3
## 12       105   9
## 13       106   3
## 14       108   7
## 15       110  58
```

```
## 16       112  14
## 17       113   1
## 18       114   2
## 19       115  19
## 20       116   2
## 21       117   1
## 22       118  10
## 23       120 132
## 24       122  12
## 25       123   2
## 26       124  12
## 27       125  29
## 28       126   7
## 29       127   1
## 30       128  18
## 31       129   1
## 32       130 118
## 33       131   4
## 34       132  17
## 35       133   6
## 36       134  11
## 37       135  20
## 38       136  13
## 39       137   5
## 40       138  17
## 41       139   5
## 42       140 107
## 43       141   3
## 44       142  11
## 45       143   2
## 46       144   8
## 47       145  18
## 48       146   4
## 49       148   2
## 50       150  55
## 51       152   7
## 52       154   3
## 53       155   8
## 54       156   2
## 55       158   4
## 56       160  50
## 57       164   1
## 58       165   2
## 59       170  14
## 60       172   2
## 61       174   1
## 62       178   3
## 63       180  12
## 64       185   1
## 65       190   2
## 66       192   1
## 67       200   4
```

```
heart %>%
  count(Age)
```

```
##    Age  n
## 1   28  1
## 2   29  3
## 3   30  1
## 4   31  2
## 5   32  5
## 6   33  2
## 7   34  7
## 8   35 11
## 9   36  6
## 10  37 11
## 11  38 16
## 12  39 15
## 13  40 13
## 14  41 24
## 15  42 18
## 16  43 24
## 17  44 19
## 18  45 18
## 19  46 24
## 20  47 19
## 21  48 31
## 22  49 21
## 23  50 25
## 24  51 35
## 25  52 36
## 26  53 33
## 27  54 51
## 28  55 41
## 29  56 38
## 30  57 38
## 31  58 42
## 32  59 35
## 33  60 32
## 34  61 31
## 35  62 35
## 36  63 30
## 37  64 22
## 38  65 21
## 39  66 13
## 40  67 15
## 41  68 10
## 42  69 13
## 43  70  7
## 44  71  5
## 45  72  4
## 46  73  1
## 47  74  7
## 48  75  3
## 49  76  2
## 50  77  2
```

```
heart %>%
  count(MaxHR)
```

```
##     MaxHR  n
## 1      60  1
## 2      63  1
## 3      67  1
## 4      69  1
## 5      70  1
## 6      71  1
## 7      72  2
## 8      73  1
## 9      77  1
## 10     78  1
## 11     80  2
## 12     82  3
## 13     83  1
## 14     84  3
## 15     86  4
## 16     87  1
## 17     88  2
## 18     90  3
## 19     91  1
## 20     92  6
## 21     93  2
## 22     94  4
## 23     95  2
## 24     96  7
## 25     97  3
## 26     98  9
## 27     99  7
## 28    100 14
## 29    102  4
## 30    103  4
## 31    104  2
## 32    105 11
## 33    106  5
## 34    107  1
## 35    108  8
## 36    109  5
## 37    110 23
## 38    111  5
## 39    112 13
## 40    113  5
## 41    114  6
## 42    115 16
## 43    116  9
## 44    117  6
## 45    118 12
## 46    119  5
## 47    120 36
## 48    121  5
## 49    122 20
## 50    123  7
```

```
## 51    124  9
## 52    125 21
## 53    126 12
## 54    127  8
## 55    128 14
## 56    129  4
## 57    130 33
## 58    131  7
## 59    132 11
## 60    133  5
## 61    134  6
## 62    135 15
## 63    136  6
## 64    137  7
## 65    138 14
## 66    139  6
## 67    140 41
## 68    141  6
## 69    142 14
## 70    143 10
## 71    144 13
## 72    145 14
## 73    146  6
## 74    147  5
## 75    148 11
## 76    149  6
## 77    150 43
## 78    151  5
## 79    152 11
## 80    153  5
## 81    154 12
## 82    155 14
## 83    156 10
## 84    157  7
## 85    158  8
## 86    159  5
## 87    160 25
## 88    161  7
## 89    162 13
## 90    163 10
## 91    164  4
## 92    165 11
## 93    166  5
## 94    167  2
## 95    168  8
## 96    169  6
## 97    170 20
## 98    171  4
## 99    172 10
## 100   173  7
## 101   174  7
## 102   175 10
## 103   176  2
## 104   177  1
```

```
## 105   178  6
## 106   179  6
## 107   180 10
## 108   181  2
## 109   182  6
## 110   184  4
## 111   185  4
## 112   186  2
## 113   187  1
## 114   188  2
## 115   190  2
## 116   192  1
## 117   194  1
## 118   195  1
## 119   202  1
```

```
heart %>%
  count(Oldpeak)
```

```
##    Oldpeak   n
## 1     -2.6   1
## 2     -2.0   1
## 3     -1.5   1
## 4     -1.1   1
## 5     -1.0   2
## 6     -0.9   1
## 7     -0.8   1
## 8     -0.7   1
## 9     -0.5   2
## 10    -0.1   2
## 11     0.0 368
## 12     0.1  14
## 13     0.2  22
## 14     0.3  11
## 15     0.4  11
## 16     0.5  19
## 17     0.6  14
## 18     0.7   7
## 19     0.8  16
## 20     0.9   4
## 21     1.0  86
## 22     1.1   7
## 23     1.2  26
## 24     1.3   7
## 25     1.4  18
## 26     1.5  53
## 27     1.6  16
## 28     1.7   6
## 29     1.8  17
## 30     1.9   7
## 31     2.0  76
## 32     2.1   2
## 33     2.2   5
## 34     2.3   2
## 35     2.4   4
```

```
## 36     2.5  16
## 37     2.6   7
## 38     2.8   7
## 39     2.9   1
## 40     3.0  28
## 41     3.1   1
## 42     3.2   2
## 43     3.4   3
## 44     3.5   2
## 45     3.6   4
## 46     3.7   1
## 47     3.8   1
## 48     4.0   8
## 49     4.2   2
## 50     4.4   1
## 51     5.0   1
## 52     5.6   1
## 53     6.2   1
```

```
heart %>%
  count(ST_Slope)
```

```
##   ST_Slope   n
## 1     Down  63
## 2     Flat 460
## 3       Up 395
```

```
heart %>%
  count(HeartDisease)
```

```
##   HeartDisease   n
## 1            0 410
## 2            1 508
```

```
heart <- heart %>%
  filter(Cholesterol != 0) %>%
  filter(RestingBP!=0) %>%
  mutate(Sex = as.factor(Sex)) %>%
  mutate(ChestPainType= as.factor(ChestPainType)) %>%
  mutate(RestingECG= as.factor(RestingECG)) %>%
  mutate(ExerciseAngina = as.factor(ExerciseAngina)) %>%
  mutate(ST_Slope = as.factor(ST_Slope)) %>%
  mutate(HeartDisease = as.factor(HeartDisease))
```

```
#set.seed(1)
#x <- model.matrix(HeartDisease~., heart)[,-1]
#y <- heart$HeartDisease

#svmfit <- svm(y ~ ., data = heart, kernel = "linear", cost = 10, scale = FALSE)
#summary(svmfit)
```

The code below creates a training and test data set.

```
set.seed(1)
training <- sample(dim(heart)[1], 522)
train <- heart[training, ]
test <- heart[-training, ]
```

The code below fits a support vector classifier.

```
svm.fit <- svm(HeartDisease ~., data = train, kernel = "linear", cost=0.01)
summary(svm.fit)
```

```
##
## Call:
## svm(formula = HeartDisease ~ ., data = train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  295
##
##  ( 148 147 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

There are 295 Support Vectors, 148 in Class 0 and 147 in Class 1.

```
svm.pred <- predict(svm.fit, test)
wrong <- 0
for (i in 1:length(svm.pred)){
  if (svm.pred[i]!=test$HeartDisease[i]){
    wrong = wrong + 1
  }
}
print(wrong/length(test$HeartDisease))
```

```
## [1] 0.1428571
```

The test error rate is 14.3%.

The code below uses CV to find the optimal cost for an svm with a linear kernel.

```
set.seed(1)
tune.out = tune(svm, HeartDisease ~ ., data = train, kernel = "linear", ranges = list(cost = 10^seq(-2,
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  5.623413
##
## - best performance: 0.1398403
##
```

```
## - Detailed performance results:
##           cost     error dispersion
## 1   0.01000000 0.1532656 0.04594129
## 2   0.01778279 0.1532656 0.04021823
## 3   0.03162278 0.1571118 0.04398593
## 4   0.05623413 0.1417634 0.03134359
## 5   0.10000000 0.1474964 0.02842013
## 6   0.17782794 0.1417271 0.02546022
## 7   0.31622777 0.1436502 0.02558931
## 8   0.56234133 0.1474601 0.03084972
## 9   1.00000000 0.1493832 0.03187442
## 10  1.77827941 0.1417634 0.03134359
## 11  3.16227766 0.1436865 0.03011101
## 12  5.62341325 0.1398403 0.03110896
## 13 10.00000000 0.1398403 0.03110896
```

The optimal cost is 5.623.

The code below calculates the test error using the optimal cost.

```
svm.fit = svm(HeartDisease ~ ., kernel = "linear", data = train, cost = tune.out$best.parameters$cost)

svm.pred <- predict(svm.fit, test)

wrong <- 0
for (i in 1:length(svm.pred)){
  if (svm.pred[i]!=test$HeartDisease[i]){
    wrong = wrong + 1
  }
}
print(wrong/length(test$HeartDisease))
```

```
## [1] 0.1205357
```

The error rate is 12.05%.

END OF LINEAR

BEGINNING OF RADIAL

The code below fits a support vector classifier.

```
svm.fitradial <- svm(HeartDisease ~., data = train, kernel = "radial")
summary(svm.fitradial)
```

```
##
## Call:
## svm(formula = HeartDisease ~ ., data = train, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  233
##
##  ( 119 114 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

There are 233 Support Vectors, with 119 in 0 and 114 in 1.

```
svm.predradial <- predict(svm.fitradial, test)

wrong <- 0
for (i in 1:length(svm.predradial)){
  if (svm.predradial[i]!=test$HeartDisease[i]){
    wrong = wrong + 1
  }
}
print(wrong/length(test$HeartDisease))
```

```
## [1] 0.1205357
```

The test error is 12.05%.

```
set.seed(1)
tune.out2 = tune(svm, HeartDisease ~ ., data = train, kernel = "radial", ranges = list(cost = 10^seq(-2
summary(tune.out2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  5.623413
##
## - best performance: 0.1399129
##
## - Detailed performance results:
##           cost     error dispersion
## 1   0.01000000 0.5171263 0.07196263
## 2   0.01778279 0.2298621 0.05191857
## 3   0.03162278 0.1800798 0.04793246
## 4   0.05623413 0.1628084 0.04793032
## 5   0.10000000 0.1513788 0.04653122
## 6   0.17782794 0.1513788 0.04740609
## 7   0.31622777 0.1551887 0.04380499
## 8   0.56234133 0.1475327 0.03145408
## 9   1.00000000 0.1437591 0.03657071
## 10  1.77827941 0.1418723 0.03554973
## 11  3.16227766 0.1456821 0.04276003
## 12  5.62341325 0.1399129 0.04156757
## 13 10.00000000 0.1456821 0.04276003
```

The optimal cost is 5.623. The code below finds the test error of a radial kernel SVM using the optimal cost.

```r
svm.radial3 = svm(HeartDisease ~ ., data = train, kernel = "radial", cost = tune.out2$best.parameters$c

svm.predradial <- predict(svm.radial3, test)

wrong <- 0
for (i in 1:length(svm.predradial)){
  if (svm.predradial[i]!=test$HeartDisease[i]){
    wrong = wrong + 1
  }
}
print(wrong/length(test$HeartDisease))
```

```
## [1] 0.1205357
```

The test error is 12.05%.

END OF RADIAL

BEGINNING OF POLYNOMIAL

```r
svm.fitpoly <- svm(HeartDisease ~., data = train, kernel = "poly", degree = 2)
summary(svm.fitpoly)
```

```
##
## Call:
## svm(formula = HeartDisease ~ ., data = train, kernel = "poly", degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  279
##
##  ( 139 140 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

There are 279 Support Vectors, 139 0 and 140 1.

```r
svm.predpoly <- predict(svm.fitpoly, test)

wrong <- 0
for (i in 1:length(svm.predpoly)){
  if (svm.predpoly[i]!=test$HeartDisease[i]){
    wrong = wrong + 1
  }
}
print(wrong/length(test$HeartDisease))
```

```
## [1] 0.1473214
```

The test error is 14.7%.

The code below find teh optimal cost

```
set.seed(1)
tune.out = tune(svm, HeartDisease ~ ., data = train, kernel = "poly", degree = 2,ranges = list(cost = 1
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##      cost
##  1.778279
##
## - best performance: 0.1456821
##
## - Detailed performance results:
##           cost     error dispersion
## 1   0.01000000 0.5171263 0.07196263
## 2   0.01778279 0.5171263 0.07196263
## 3   0.03162278 0.4518505 0.07328965
## 4   0.05623413 0.2813498 0.06687009
## 5   0.10000000 0.2183962 0.03164476
## 6   0.17782794 0.1781930 0.04068122
## 7   0.31622777 0.1628447 0.03866025
## 8   0.56234133 0.1532656 0.03949555
## 9   1.00000000 0.1494920 0.03938606
## 10  1.77827941 0.1456821 0.03872582
## 11  3.16227766 0.1533745 0.04266201
## 12  5.62341325 0.1572206 0.04881518
## 13 10.00000000 0.1725327 0.05379036
```

The optimal cost is 1.778.

The code below calculates the test error using the optimal cost.

```
svm.poly3 = svm(HeartDisease ~ ., data = train, kernel = "poly", degree = 2, cost = tune.out$best.param

test.pred3 = predict(svm.poly3, test)

wrong <- 0
for (i in 1:length(test.pred3)){
  if (test.pred3[i]!=test$HeartDisease[i]){
    wrong = wrong + 1
  }
}
print(wrong/length(test$HeartDisease))
```

```
## [1] 0.1294643
```

The test error is 12.95%.

CONCLUSION
Linear Kernel w/Optimal Cost: 12.05%
Radial Kernel w/Optimal Cost: 12.05%
Polynomial Kernel w/ Optimal Cost: 12.95%