Kernel Methods Classification

David Park

2022-10-25

```
library(e1071)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
set.seed(1234)
```

Extract 10k observations from dataset

```
df <- read.csv("C:\\School\\CS 4375 Machine Learning\\Kernel Ensemble Methods\\Invistico_Airline.csv")
df <- sample_n(df, 10000, replace=FALSE)
df$satisfaction <- as.factor(df$satisfaction)
```

Combine columns that give rating into one

```
df$rating.mean <- as.numeric(apply(df[, 8:21], 1, sum)) / 14
df <- df[, c(1,4,24)]
str(df)
```

```
## 'data.frame':    10000 obs. of  3 variables:
##  $ satisfaction: Factor w/ 2 levels "dissatisfied",..: 2 2 1 2 1 2 2 2 1 2 ...
##  $ Age         : int  32 43 66 69 10 36 42 46 41 26 ...
##  $ rating.mean : num  3 3.14 2.79 3.57 3 ...
```

```
head(df)
```

```
##   satisfaction Age rating.mean
## 1    satisfied  32    3.000000
## 2    satisfied  43    3.142857
```

```
## 3 dissatisfied  66    2.785714
## 4    satisfied  69    3.571429
## 5 dissatisfied  10    3.000000
## 6    satisfied  36    3.214286
```

Divide train/test

```
spec <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df), nrow(df)*cumsum(c(0,spec)), labels=names(spec)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

Explore the training data statistically and graphically

```
str(train)
```

```
## 'data.frame':    6000 obs. of  3 variables:
##  $ satisfaction: Factor w/ 2 levels "dissatisfied",..: 2 2 1 2 2 2 1 2 1 1 ...
##  $ Age         : int  32 43 66 69 36 26 51 46 37 45 ...
##  $ rating.mean : num  3 3.14 2.79 3.57 3.21 ...
```

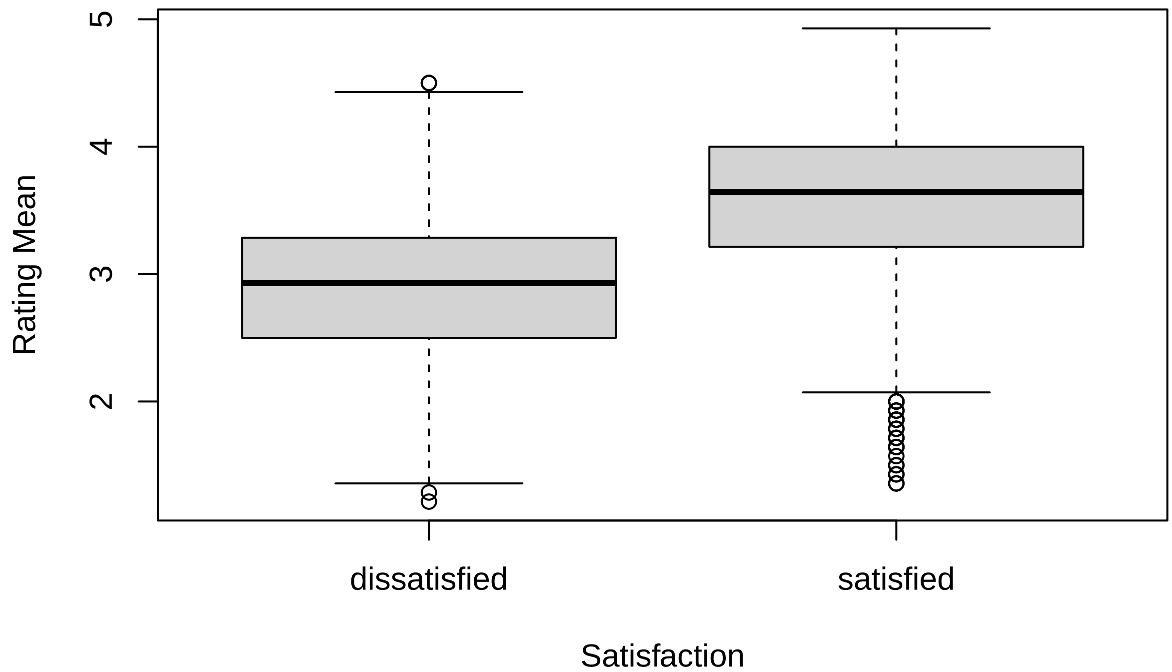
```
dim(train)
```

```
## [1] 6000    3
```

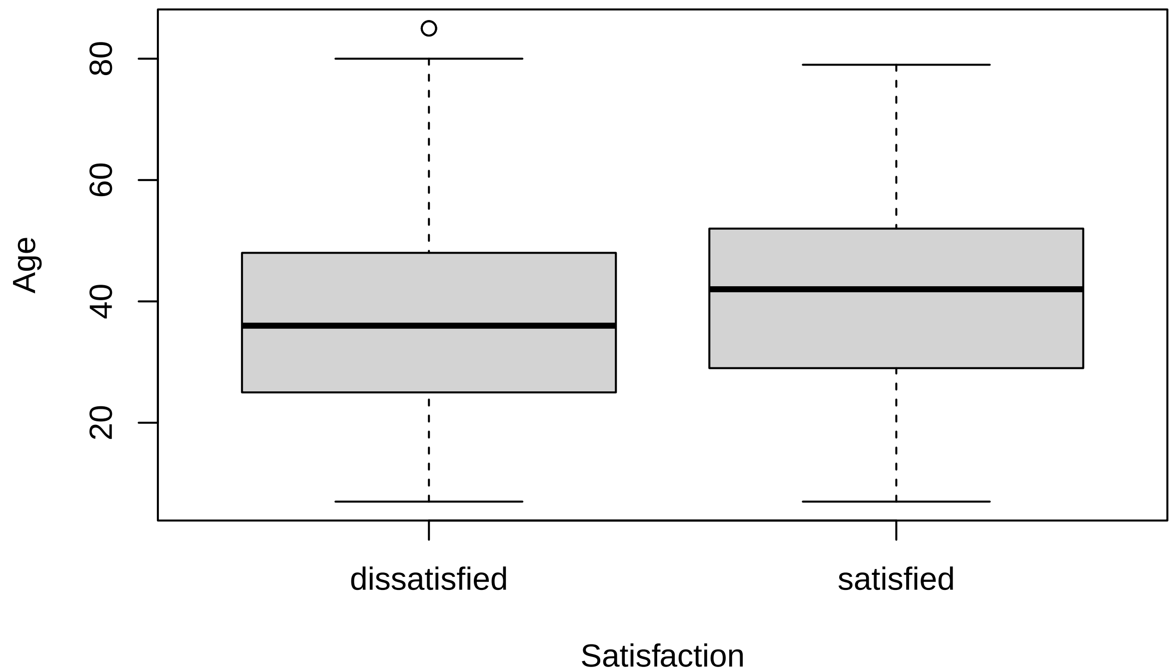
```
summary(train)
```

```
##        satisfaction       Age         rating.mean
##  dissatisfied:2775   Min.   : 7.00   Min.   :1.214
##  satisfied   :3225   1st Qu.:27.00   1st Qu.:2.786
##                      Median :39.00   Median :3.286
##                      Mean   :39.14   Mean   :3.294
##                      3rd Qu.:51.00   3rd Qu.:3.786
##                      Max.   :85.00   Max.   :4.929
```

```
plot(train$satisfaction, train$rating.mean, xlab = "Satisfaction", ylab = "Rating Mean")
```




```r
plot(train$satisfaction, train$Age, xlab = "Satisfaction", ylab = "Age")
```



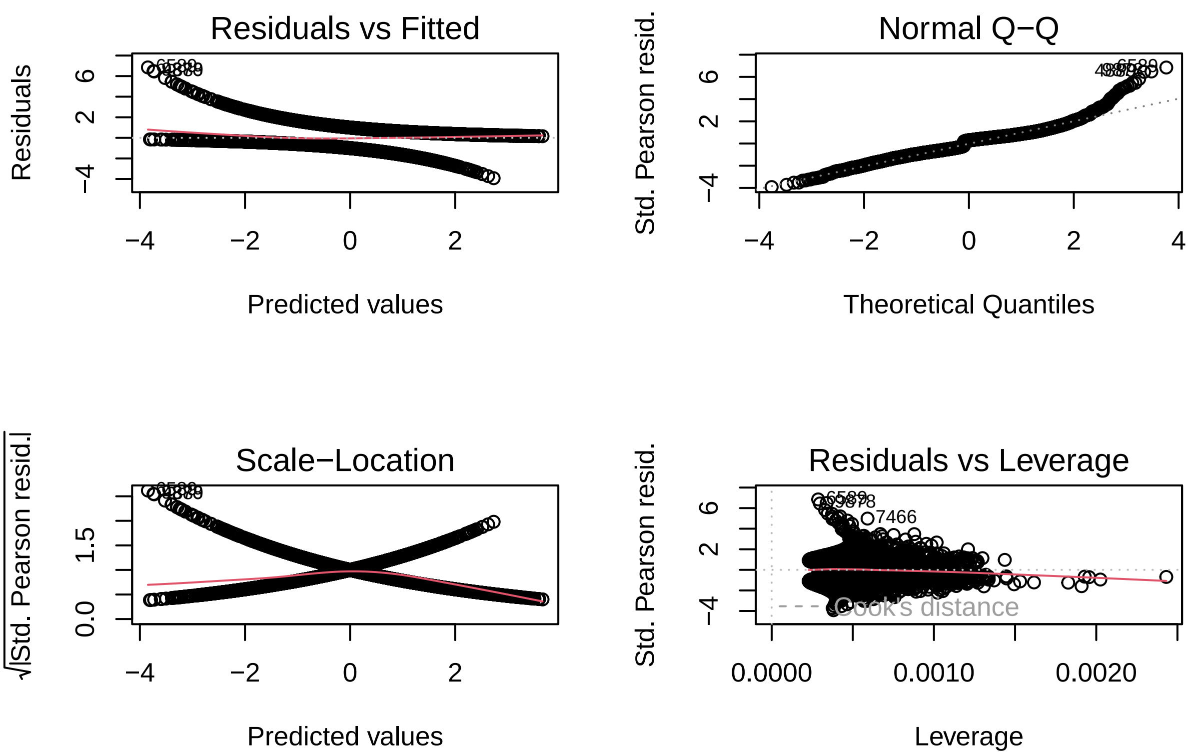
Logistic regression for some baseline

```
glm <- glm(satisfaction~., data=train, family=binomial)
summary(glm)
```

```
##
## Call:
## glm(formula = satisfaction ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3644  -0.8993   0.3998   0.8531   2.7814
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.764123   0.202868 -33.343  < 2e-16 ***
## Age          0.013237   0.001985   6.667 2.61e-11 ***
## rating.mean  1.954379   0.056623  34.515  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8284.0  on 5999  degrees of freedom
```

```
## Residual deviance: 6474.4  on 5997  degrees of freedom
## AIC: 6480.4
##
## Number of Fisher Scoring iterations: 4
```

```r
par(mfrow=c(2,2))
plot(glm)
```




```r
prob <- predict(glm, newdata=test, type="response")
pred <- ifelse(prob>0.5, 2, 1)
acc <- mean(pred==as.integer(test$satisfaction))
acc
```

```
## [1] 0.7375
```

Linear Kernel SVM classification

```r
svm_linear <- svm(satisfaction~., data=train, kernel="linear", cost=10, scale=TRUE)
summary(svm_linear)
```

```
##
## Call:
```

```
## svm(formula = satisfaction ~ ., data = train, kernel = "linear",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  10
##
## Number of Support Vectors:  3722
##
##  ( 1862 1860 )
##
##
## Number of Classes:  2
##
## Levels:
##  dissatisfied satisfied
```

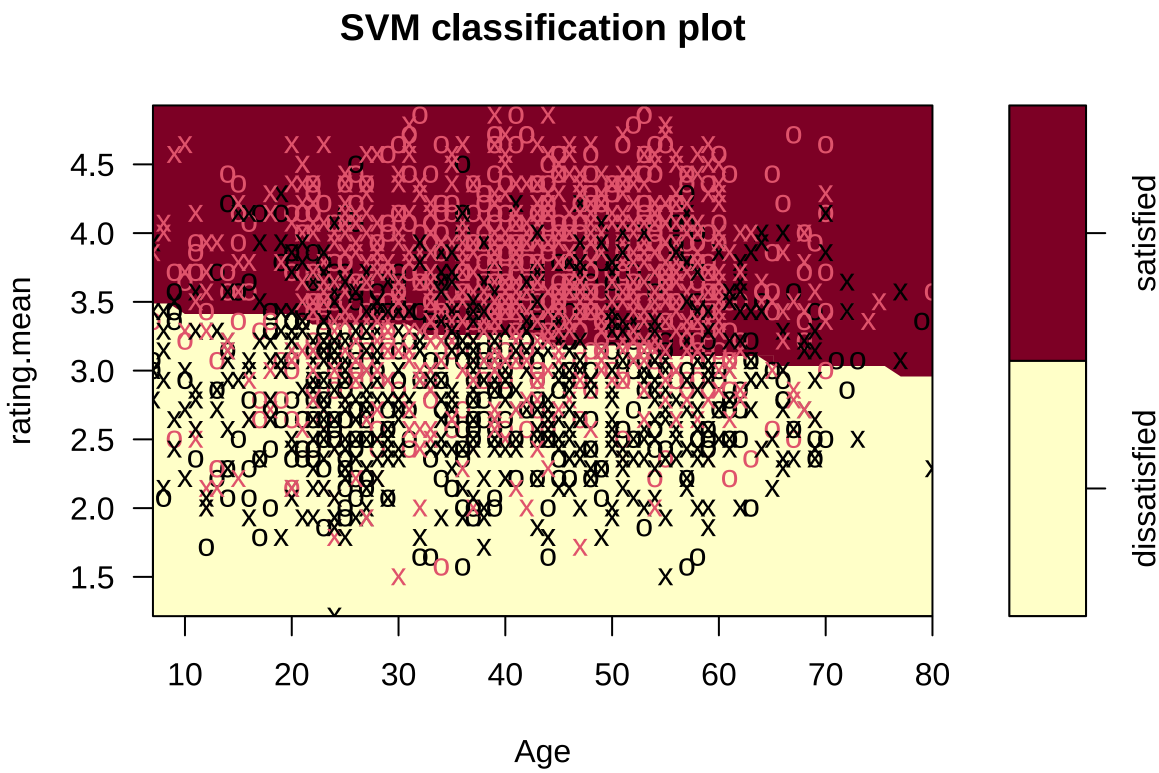
```
pred <- predict(svm_linear, newdata=test)
table(pred, test$satisfaction)
```

```
##
## pred           dissatisfied satisfied
##   dissatisfied          598       248
##   satisfied             273       881
```

```
mean(pred==test$satisfaction)
```

```
## [1] 0.7395
```

```
plot(svm_linear, test, rating.mean ~ Age)
```



Tune linear kernel SVM

```
tune_svm_linear <- tune(svm, satisfaction~., data=vald, kernel="linear", ranges=list(cost=c(0.001, 0.01
summary(tune_svm_linear)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.2545
##
## - Detailed performance results:
##    cost  error dispersion
## 1 1e-03 0.2730 0.03056868
## 2 1e-02 0.2545 0.02763351
## 3 1e-01 0.2565 0.02906411
## 4 1e+00 0.2545 0.02910231
## 5 5e+00 0.2550 0.02915476
## 6 1e+01 0.2550 0.02905933
```

```
## 7 1e+02 0.2555 0.02910231
```

Evaluate linear kernel SVM with best model

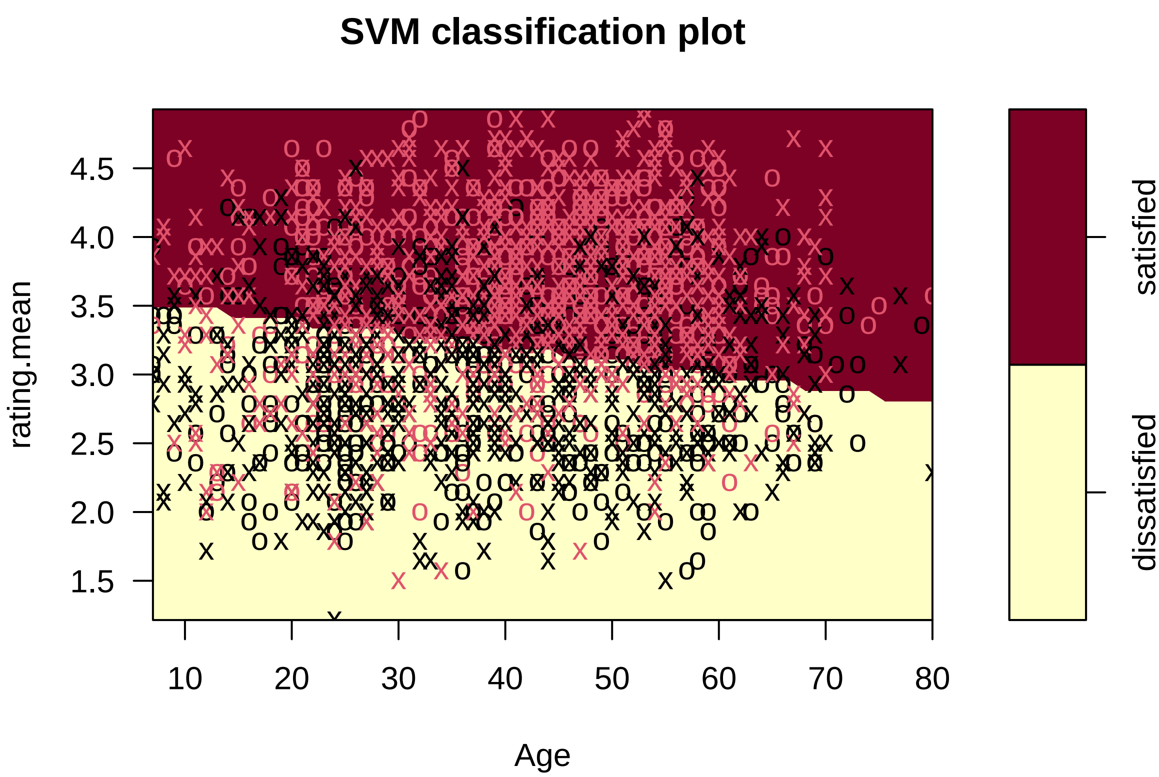
```r
pred <- predict(tune_svm_linear$best.model, newdata=test)
table(pred, test$satisfaction)
```

```
##
## pred           dissatisfied satisfied
##   dissatisfied          564       225
##   satisfied             307       904
```

```r
mean(pred==test$satisfaction)
```

```
## [1] 0.734
```

```r
plot(tune_svm_linear$best.model, test, rating.mean ~ Age)
```



Polynomial Kernel SVM Regression

```r
svm_polynomial <- svm(satisfaction~., data=train, kernel="polynomial", cost=10, scale=TRUE)
summary(svm_polynomial)
```

```
##
## Call:
## svm(formula = satisfaction ~ ., data = train, kernel = "polynomial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  4388
##
##  ( 2195 2193 )
##
##
## Number of Classes:  2
##
## Levels:
##  dissatisfied satisfied
```

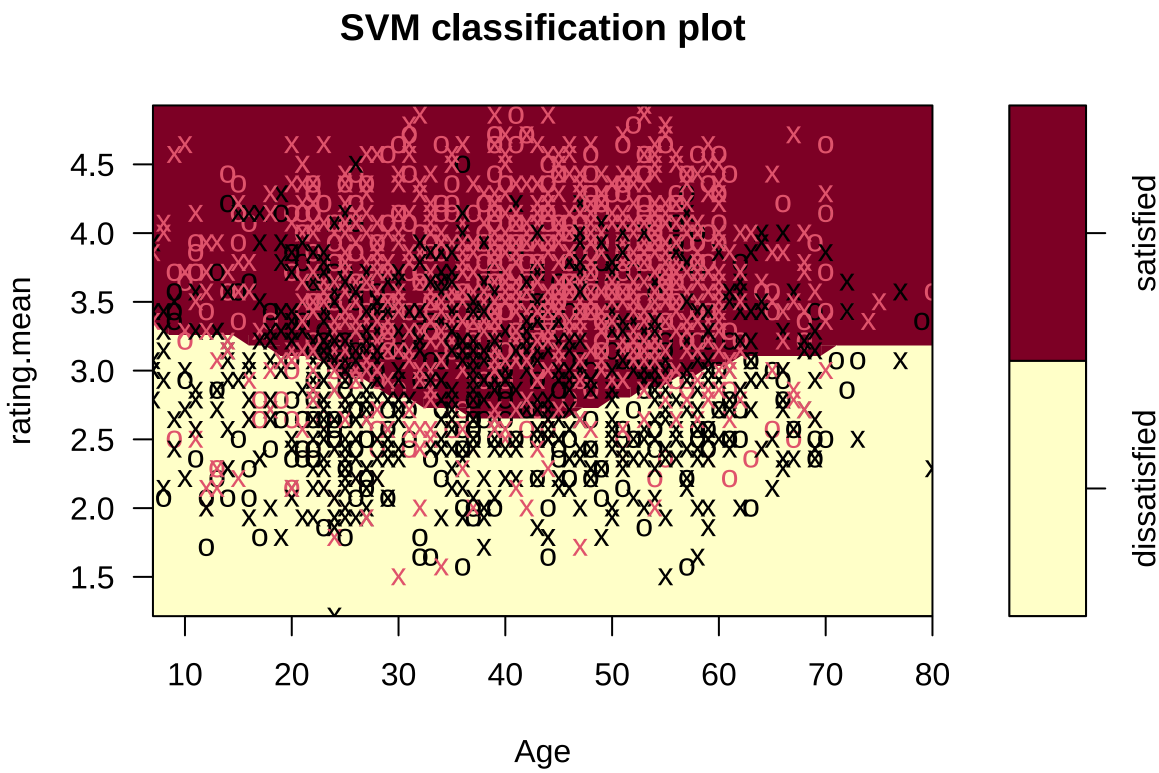
```r
pred <- predict(svm_polynomial, newdata=test)
table(pred, test$satisfaction)
```

```
##
## pred           dissatisfied satisfied
##   dissatisfied          399       117
##   satisfied             472      1012
```

```r
mean(pred==test$satisfaction)
```

```
## [1] 0.7055
```

```r
plot(svm_polynomial, test, rating.mean ~ Age)
```



Tune polynomial kernel SVM

```
tune_svm_poly <- tune(svm, satisfaction~., data=vald, kernel="polynomial", ranges=list(cost=c(0.001, 0.0
summary(tune_svm_poly)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.302
##
## - Detailed performance results:
##     cost  error dispersion
## 1 1e-03 0.3975 0.04124790
## 2 1e-02 0.3260 0.04081122
## 3 1e-01 0.3020 0.03537105
## 4 1e+00 0.3020 0.03417276
## 5 5e+00 0.3020 0.03417276
## 6 1e+01 0.3020 0.03417276
```

```
## 7 1e+02 0.3020 0.03417276
```

Evaluate polynomial kernel SVM

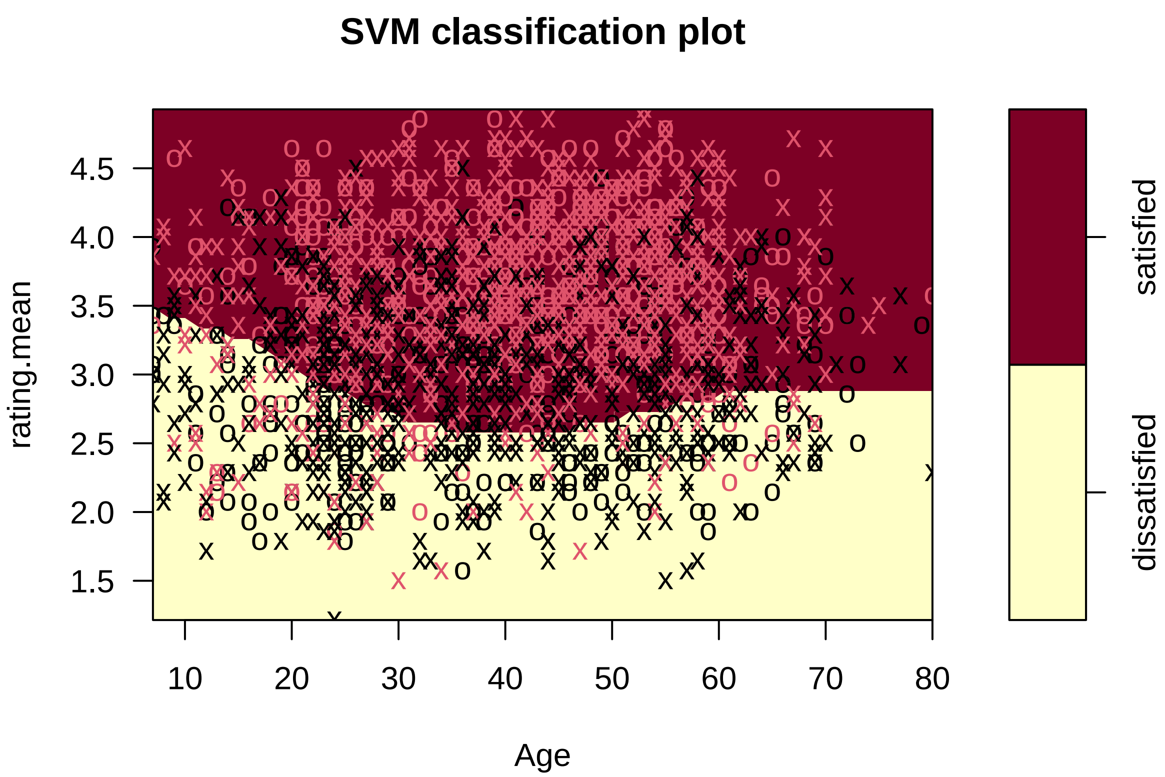
```
pred <- predict(tune_svm_poly$best.model, newdata=test)
table(pred, test$satisfaction)
```

```
##
## pred            dissatisfied satisfied
##   dissatisfied           350        94
##   satisfied              521      1035
```

```
mean(pred==test$satisfaction)
```

```
## [1] 0.6925
```

```
plot(tune_svm_poly$best.model, test, rating.mean ~ Age)
```



Radial Kernel SVM Regression

```r
svm_radial <- svm(satisfaction~., data=train, kernel="radial", cost=10, scale=TRUE)
summary(svm_radial)
```

```
##
## Call:
## svm(formula = satisfaction ~ ., data = train, kernel = "radial",
##     cost = 10, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  3331
##
##  ( 1666 1665 )
##
##
## Number of Classes:  2
##
## Levels:
##  dissatisfied satisfied
```

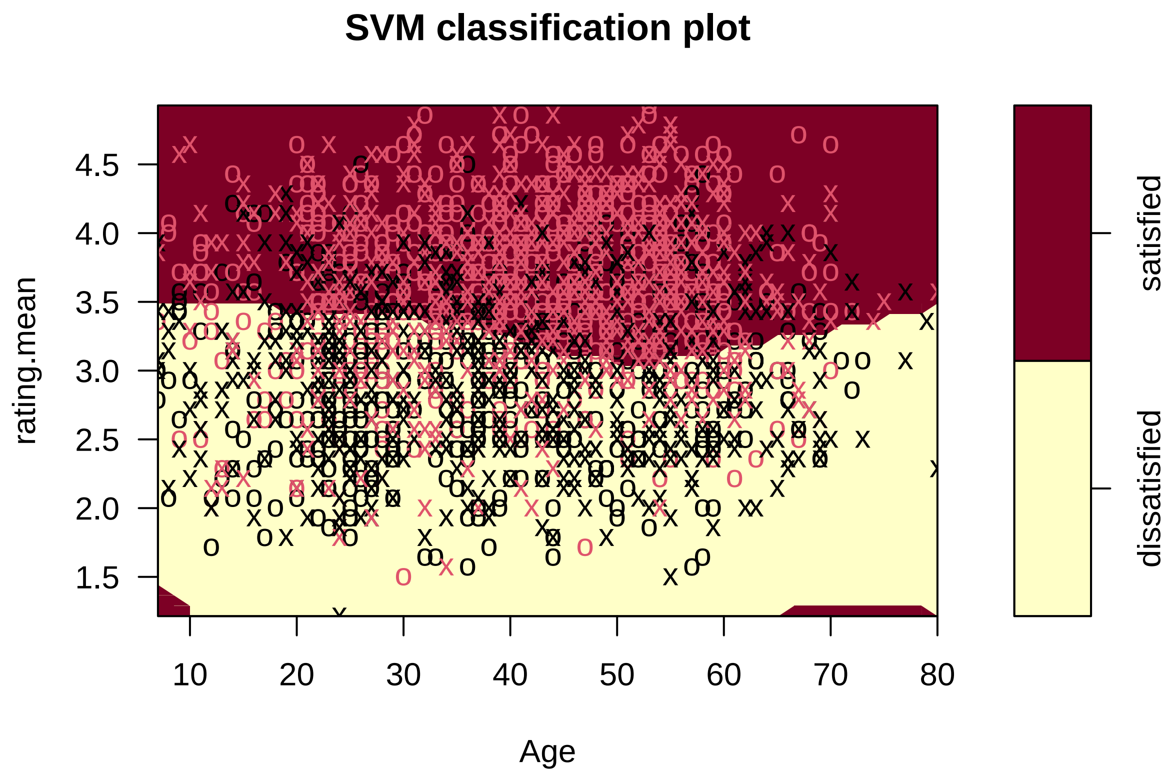
```r
pred <- predict(svm_radial, newdata=test)
table(pred, test$satisfaction)
```

```
##
## pred           dissatisfied satisfied
##   dissatisfied          611       256
##   satisfied             260       873
```

```r
mean(pred==test$satisfaction)
```

```
## [1] 0.742
```

```r
plot(svm_radial, test, rating.mean ~ Age)
```



Tune radial kernel svm

```
tune_svm_radial <- tune(svm, satisfaction~., data=vald, kernel="radial", ranges=list(cost=c(0.1, 1, 10,
summary(tune_svm_radial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   0.1   0.5
##
## - best performance: 0.2585
##
## - Detailed performance results:
##      cost gamma  error dispersion
## 1  1e-01   0.5 0.2585 0.03786306
## 2  1e+00   0.5 0.2605 0.03554731
## 3  1e+01   0.5 0.2625 0.03553168
## 4  1e+02   0.5 0.2630 0.04049691
## 5  1e+03   0.5 0.2620 0.03787406
## 6  1e-01   1.0 0.2590 0.03642344
```

```
## 7   1e+00   1.0 0.2610 0.03580813
## 8   1e+01   1.0 0.2615 0.03793635
## 9   1e+02   1.0 0.2675 0.03795099
## 10  1e+03   1.0 0.2675 0.03466106
## 11  1e-01   2.0 0.2615 0.03682164
## 12  1e+00   2.0 0.2630 0.03902990
## 13  1e+01   2.0 0.2710 0.03921451
## 14  1e+02   2.0 0.2720 0.04029061
## 15  1e+03   2.0 0.2770 0.04679744
## 16  1e-01   3.0 0.2620 0.03780065
## 17  1e+00   3.0 0.2690 0.03885872
## 18  1e+01   3.0 0.2730 0.04070217
## 19  1e+02   3.0 0.2715 0.04743709
## 20  1e+03   3.0 0.2785 0.05452064
## 21  1e-01   4.0 0.2640 0.03820995
## 22  1e+00   4.0 0.2690 0.04005552
## 23  1e+01   4.0 0.2705 0.04615493
## 24  1e+02   4.0 0.2775 0.04837642
## 25  1e+03   4.0 0.2925 0.04739022
```

Evaluate radial kernel svm with best model

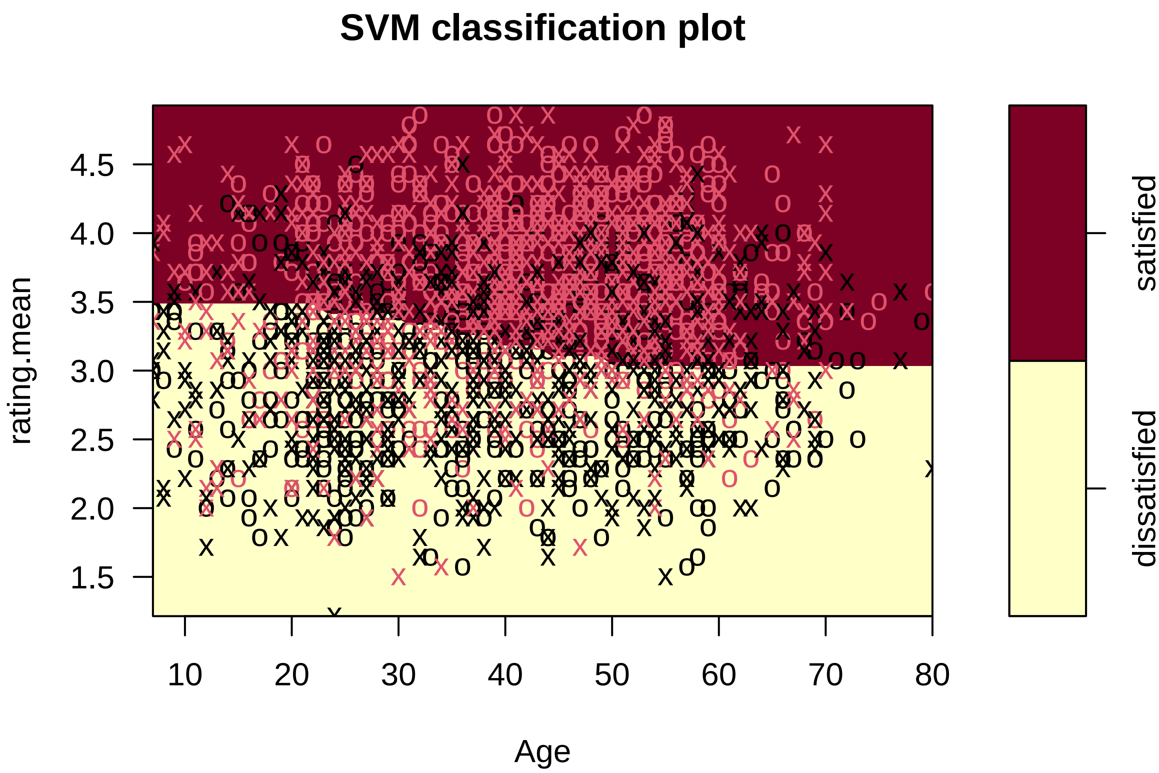
```
pred <- predict(tune_svm_radial$best.model, newdata=test)
table(pred, test$satisfaction)
```

```
##
## pred           dissatisfied satisfied
##   dissatisfied          583       242
##   satisfied             288       887
```

```
mean(pred==test$satisfaction)
```

```
## [1] 0.735
```

```
plot(tune_svm_radial$best.model, test, rating.mean ~ Age)
```



How Results Were Achieved?

For my classification data, radial kernel SVM worked the best marginally compared to linear and polynomial. This could be due to the nature of the data not being able to be separated linearly in a clean manner. However, it should be noted that the difference between my radial and linear results were very minimal. This could be the direct result of me aggregating the features that have a rating from 1 to 5 into one called the rating mean. In addition, the tuning process did not turn out as expected, again possibly due to the aggregation step that was done at the beginning.