Homework 6: Support Vector Machines

Wen Li Teng

March 8 2020

```r
set.seed(1234)
library(tidyverse)
library(rsample)
library(caret)
library(doParallel)
library(ggplot2)
library(pROC)
```

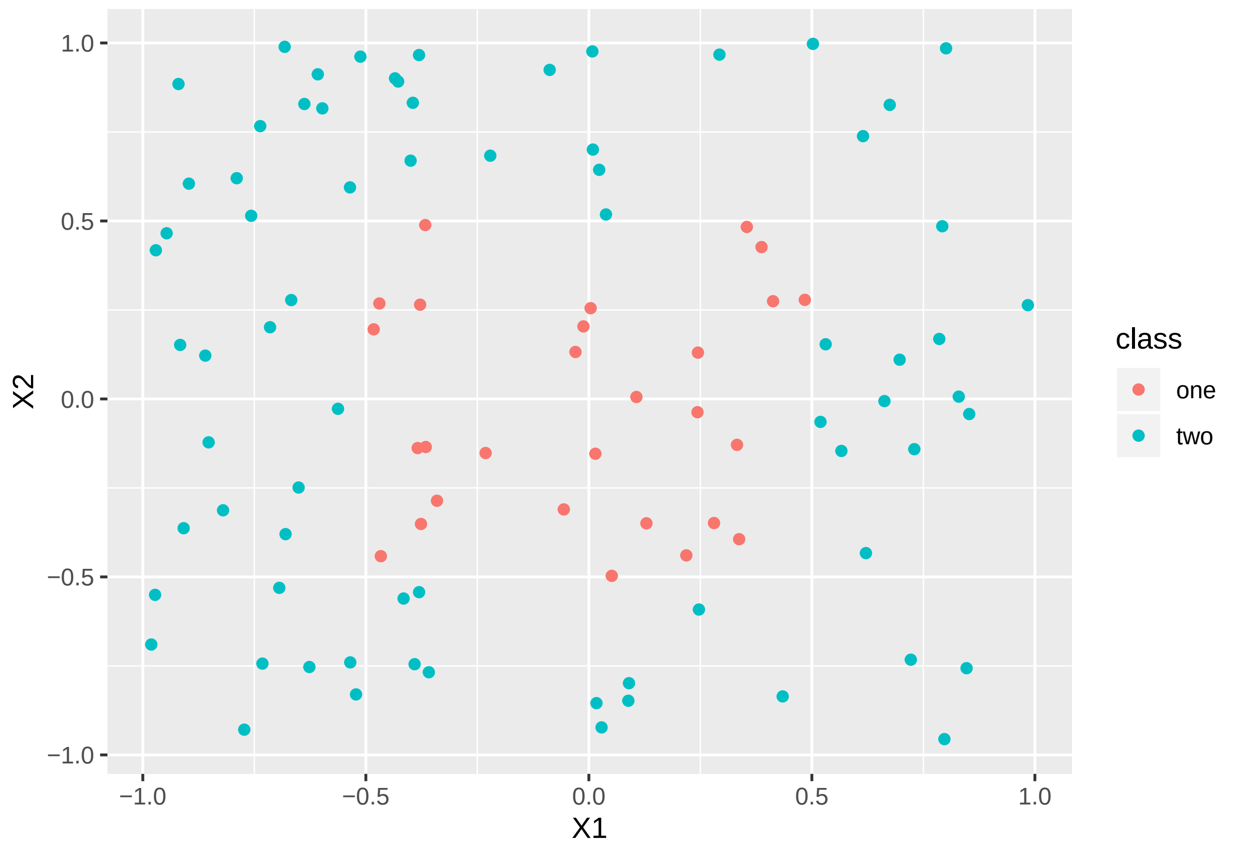
CONCEPTUAL EXERCISES

1. Non-Linear Separation

Generate a simulated two-class dataset

```r
q1_n_obs <- 100
q1_min <- -1
q1_max <- 1
q1_data <- tibble(X1 = runif(q1_n_obs, q1_min, q1_max),
               X2 = runif(q1_n_obs, q1_min, q1_max))

q1_classify <- function(X1, X2) {
  if (X1 > -0.5 & X1 < 0.5 & X2 > -0.5 & X2 < 0.5)
    output <- "one"
  else
    output <- "two"
  return(output)
}

q1_data <- q1_data %>%
  rowwise() %>%
  mutate(class = q1_classify(X1, X2))

q1_data$class <- as.factor(q1_data$class)

q1_split <- initial_split(q1_data, prop = 0.7)
q1_train <- training(q1_split)
q1_test <- testing(q1_split)
```

```
ggplot(q1_data, aes(x = X1, y = X2)) +
  geom_point(aes(color = class))
```



Performance of SVM (Radial) on Training Data

```
cv_ctrl <- trainControl(method = "cv",
                        number = 10,
                        savePredictions = "final",
                        classProbs = TRUE)

cl <- makePSOCKcluster(8)
registerDoParallel(cl)

q1_svm_radial <- train(
  class ~ .,
  data = q1_train,
  method = "svmRadial",
  trControl = cv_ctrl
)

q1_train_svm_radial_roc <- roc(predictor = q1_svm_radial$pred$one,
                      response = q1_svm_radial$pred$obs,
                      levels = rev(levels(q1_train$class)))
```

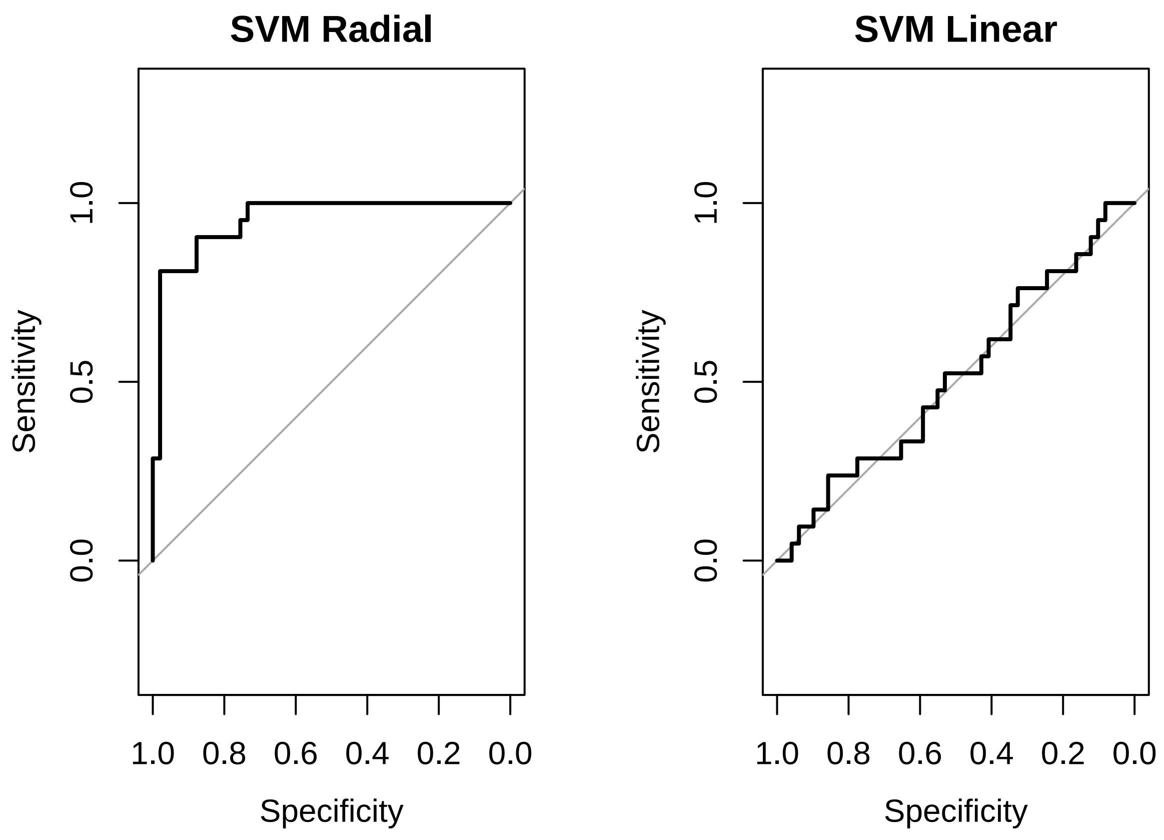
```r
q1_train_svm_radial_pred <- q1_svm_radial %>% predict(q1_train)
q1_train_svm_radial_cmat <- confusionMatrix(data = q1_train_svm_radial_pred,
                           reference = q1_train$class)
q1_train_svm_radial_err <- 1 - q1_train_svm_radial_cmat$overall["Accuracy"]
```

Performance of SVC (Linear) on Training Data

```r
q1_svm_linear <- train(
  class ~ .,
  data = q1_train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10
)

q1_train_svm_linear_roc <- roc(predictor = q1_svm_linear$pred$one,
                      response = q1_svm_linear$pred$obs,
                      levels = rev(levels(q1_train$class)))

q1_train_svm_linear_pred <- q1_svm_linear %>% predict(q1_train)
q1_train_svm_linear_cmat <- confusionMatrix(data = q1_train_svm_linear_pred,
                                           reference = q1_train$class)
q1_train_svm_linear_err <- 1 - q1_train_svm_linear_cmat$overall['Accuracy']
```

Comparison on Training Data

```r
par(mfrow=c(1,2))
plot(q1_train_svm_radial_roc, main = "SVM Radial")
plot(q1_train_svm_linear_roc, main = "SVM Linear")
```



```r
q1_train_comp <- tibble(model = c("svm radial", "svm linear"),
                        auc = c(auc(q1_train_svm_radial_roc),
                                auc(q1_train_svm_linear_roc)),
                        error = c(q1_train_svm_radial_err, q1_train_svm_linear_err))
q1_train_comp
```

```
## # A tibble: 2 x 3
##   model        auc  error
##   <chr>      <dbl>  <dbl>
## 1 svm radial 0.953 0.0714
## 2 svm linear 0.513 0.3
```

The plots and table show that SVM (Radial) outperforms SVC (Linear).

Performance of SVM (Radial) on Test Data

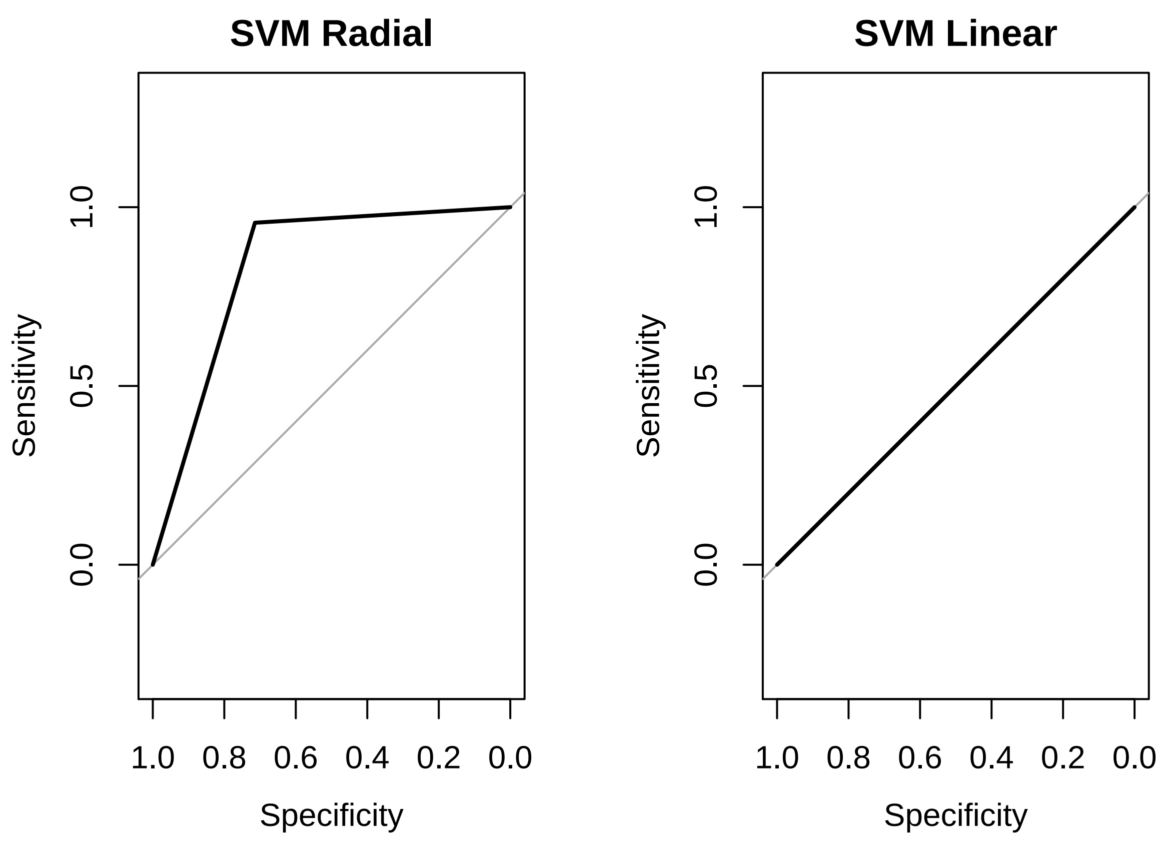
```r
q1_test_svm_radial_pred <- q1_svm_radial %>% predict(q1_test)
q1_test_svm_radial_cmat <- confusionMatrix(data = q1_test_svm_radial_pred,
                           reference = q1_test$class)
q1_test_svm_radial_err <- 1 - q1_test_svm_radial_cmat$overall["Accuracy"]

q1_test_svm_radial_roc <- roc(as.numeric(q1_test$class),
                              as.numeric(q1_test_svm_radial_pred))
```

Performance of SVC (Linear) on Test Data

```r
q1_test_svm_linear_pred <- q1_svm_linear %>% predict(q1_test)
q1_test_svm_linear_cmat <- confusionMatrix(data = q1_test_svm_linear_pred,
                                           reference = q1_test$class)
q1_test_svm_linear_err <- 1 - q1_test_svm_linear_cmat$overall['Accuracy']

q1_test_svm_linear_roc <- roc(as.numeric(q1_test$class),
                              as.numeric(q1_test_svm_linear_pred))
```

Comparison on Test Data

```r
par(mfrow=c(1,2))
plot(q1_test_svm_radial_roc, main = "SVM Radial")
plot(q1_test_svm_linear_roc, main = "SVM Linear")
```



```r
q1_test_comp <- tibble(model = c("svm radial", "svm linear"),
                 auc = c(auc(q1_test_svm_radial_roc),
                         auc(q1_test_svm_linear_roc)),
                  error = c(q1_test_svm_radial_err, q1_test_svm_linear_err))
q1_test_comp
```

```
## # A tibble: 2 x 3
##   model        auc error
##   <chr>      <dbl> <dbl>
## 1 svm radial 0.835 0.100
## 2 svm linear 0.5   0.233
```

The plots and table show that SVM (Radial) outperforms SVC (Linear).

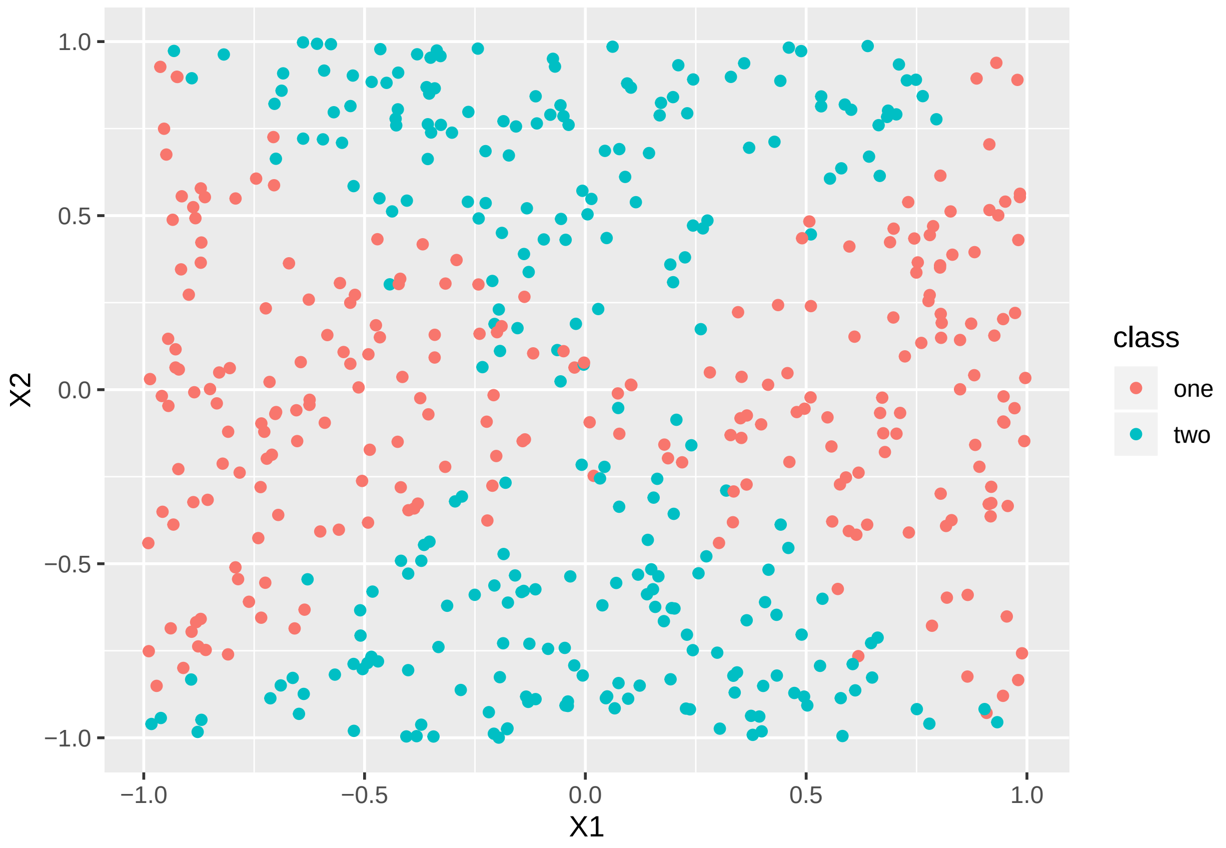
SVM VS. LOGISTIC REGRESSION

2. Generate a simulated two-class dataset with overlapping, non-linear boundary

```
q2_n_obs <- 500
q2_min <- -1
q2_max <- 1
q2_data <- tibble(X1 = runif(q2_n_obs, q2_min, q2_max),
                  X2 = runif(q2_n_obs, q2_min, q2_max),
                  perturb = rnorm(500, 0, 0.1))

q2_classify <- function(X1, X2, perturb) {
  if (1 * (X1^2 - X2^2 + perturb > 0))
    output <- "one"
  else
    output <- "two"
  return(output)
}

q2_data <- q2_data %>%
  rowwise() %>%
  mutate(class = q2_classify(X1, X2, perturb))

q2_data$class <- as.factor(q2_data$class)

q2_split <- initial_split(q2_data, prop = 0.7)
q2_train <- training(q2_split)
q2_test <- testing(q2_split)
```

3. Plot observations with colors according to class labels

```
ggplot(q2_data, aes(x = X1, y = X2)) +
  geom_point(aes(color = class))
```



4. Fit a logistic regression model using X1 and X2 as predictors

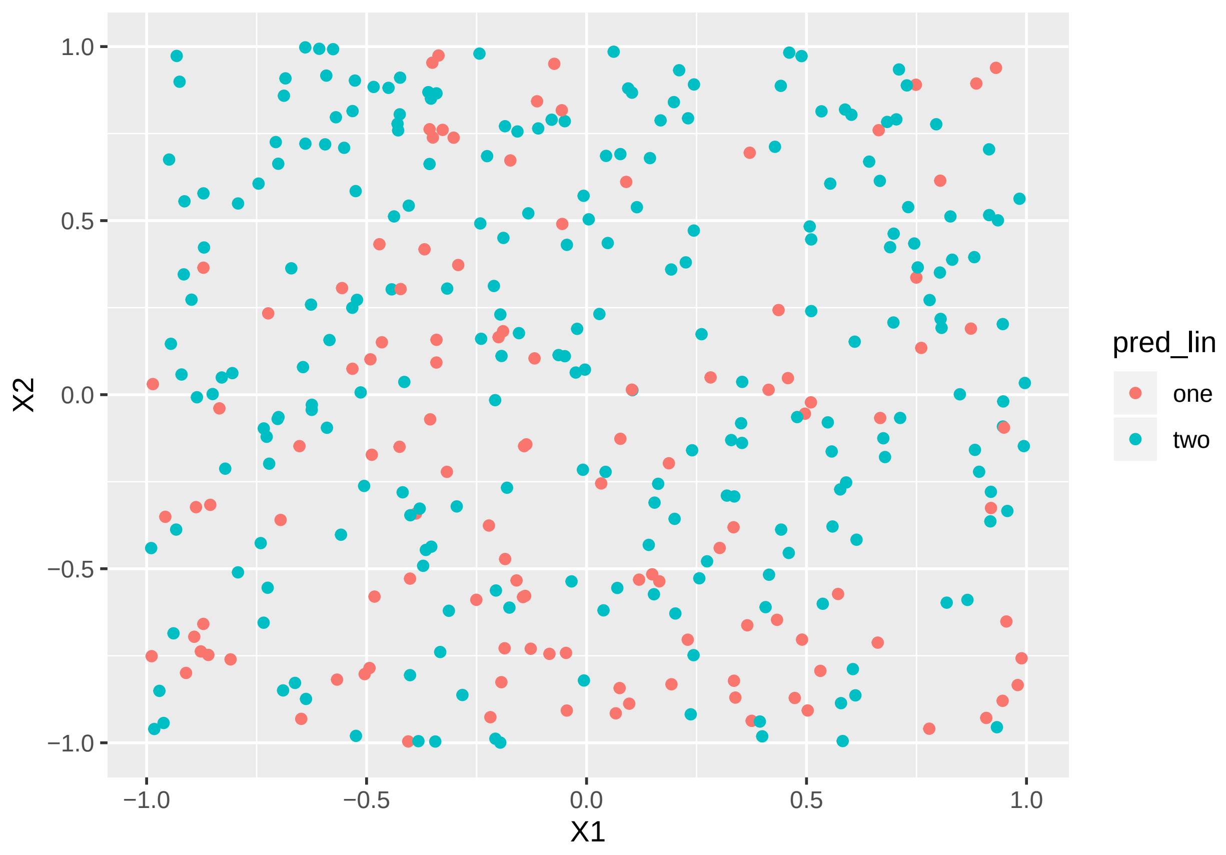
```
logit_mod_linear <- train(class ~ .,
                   data = q2_train,
                   method = "glm",
                   trControl = cv_ctrl)
```

5. Obtain a predicted class label usign linear transformation

```
q2_train <- as.data.frame(cbind(q2_train, logit_mod_linear %>% predict(q2_train)))
q2_train <- q2_train %>% rename(pred_lin = names(rev(q2_train)[1]))
```

Plot observations with colors according to class labels

```
ggplot(q2_train, aes(x = X1, y = X2)) +
  geom_point(aes(color = pred_lin))
```



6. Fit a logistic regression model using non-linear function of X1 and X2

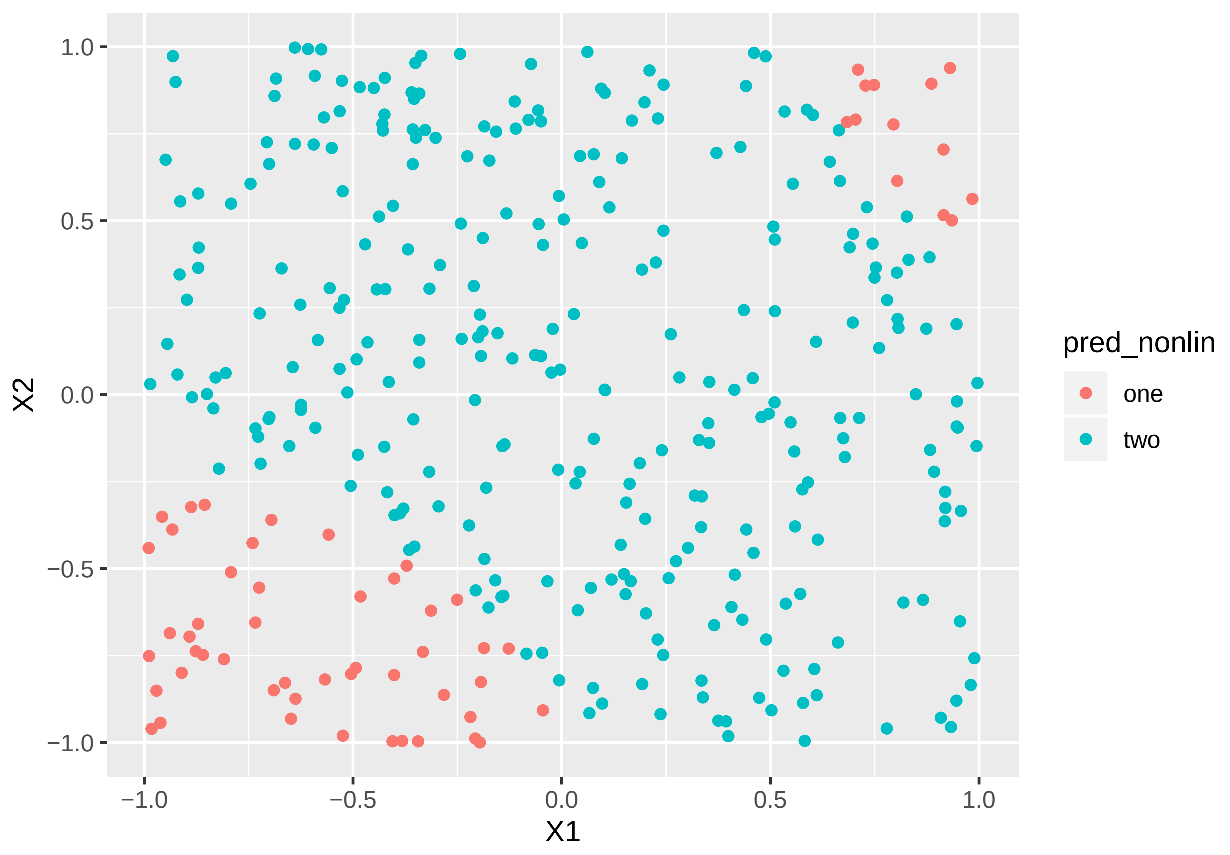
```
logit_mod_nonlinear <- train(class ~ X1^2 + X2^2 + I(X1 * X2),
                   data = q2_train,
                   method = "glm",
                   trControl = cv_ctrl)
```

7. Obtain a predicted class label using non-linear transformation

```
q2_train <- as.data.frame(cbind(q2_train, logit_mod_nonlinear %>% predict(q2_train)))
q2_train <- q2_train %>% rename(pred_nonlin = names(rev(q2_train)[1]))
```

Plot observations with colors according to class labels

```
ggplot(q2_train, aes(x = X1, y = X2)) +
  geom_point(aes(color = pred_nonlin))
```



8. Fit a SVC (Linear) to data with X1 and X2 predictors

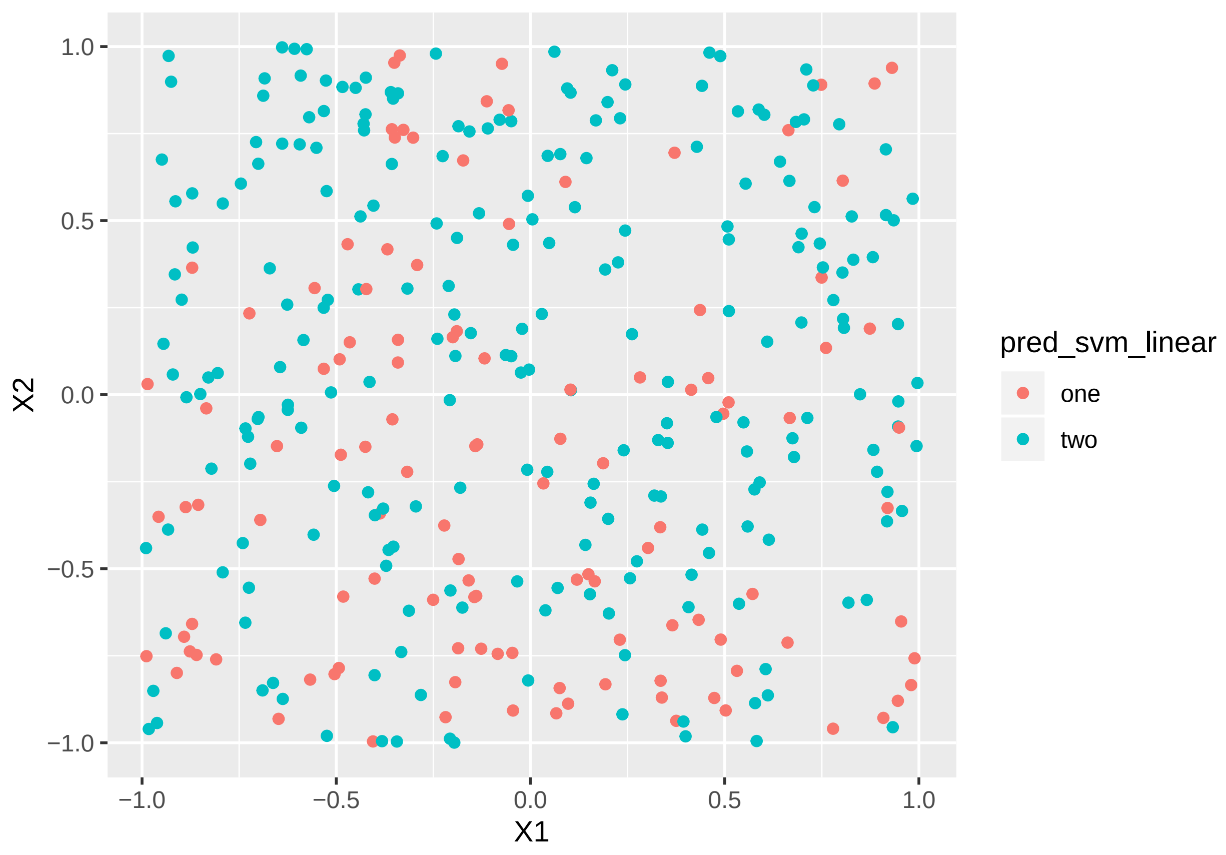
```r
svm_linear_mod <- train(
  class ~ .,
  data = q2_train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10
)
```

Obtain a class prediction for each observation

```r
q2_train <- as.data.frame(cbind(q2_train, svm_linear_mod %>% predict(q2_train)))
q2_train <- q2_train %>% rename(pred_svm_linear = names(rev(q2_train)[1]))
```

Plot observations with colors according to predicted class labels

```r
ggplot(q2_train, aes(x = X1, y = X2)) +
  geom_point(aes(color = pred_svm_linear))
```



9. Fit a SVM (Non-Linear) to data

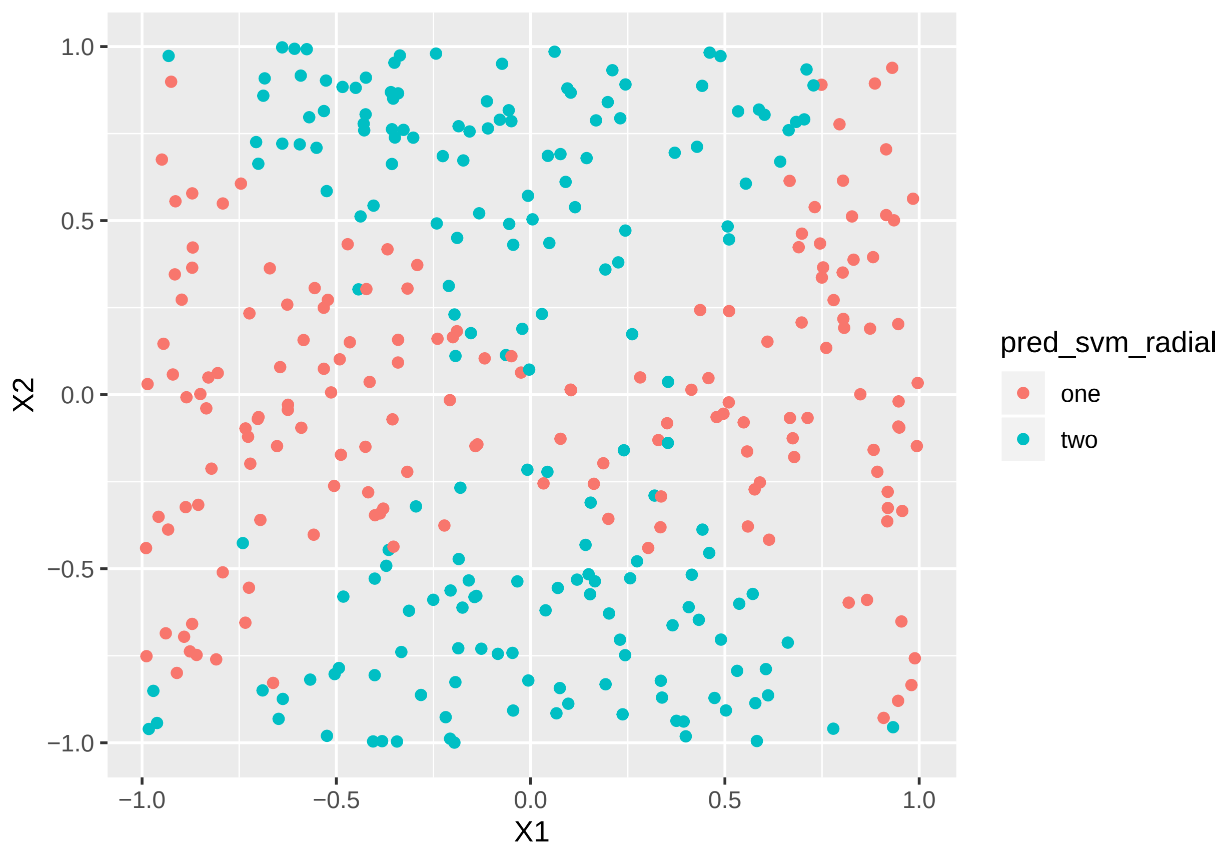
```
svm_radial_mod <- train(
  class ~ .,
  data = q2_train,
  method = "svmRadial",
  trControl = cv_ctrl
)
```

Obtain a class prediction for each observation

```
q2_train <- as.data.frame(cbind(q2_train, svm_radial_mod %>% predict(q2_train)))
q2_train <- q2_train %>% rename(pred_svm_radial = names(rev(q2_train)[1]))
```

Plot observations with colors according to predicted class labels

```
ggplot(q2_train, aes(x = X1, y = X2)) +
  geom_point(aes(color = pred_svm_radial))
```



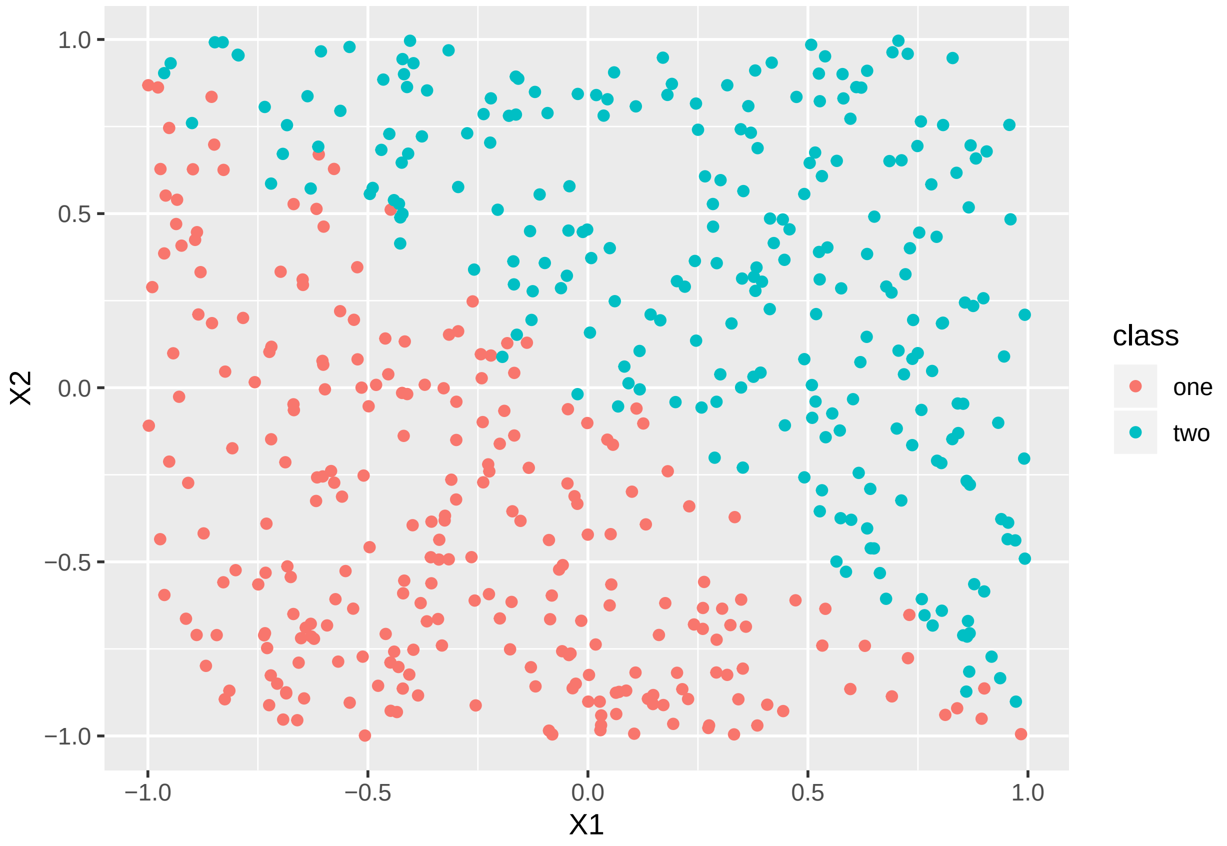
10. Discussion

The plots show that SVM (Radial) outperforms the linear logistic regression model, the non-linear logistic regression model, and SVC (Linear). SVM (Radial) outperforms the linear logistic regression model and SVC (Linear) because of the non-linear decision boundary on the original data set. SVM (Radial) also outperforms the non-linear logistic regression model because, as James et al. (2013) observe, "[w]hen the classes are well separated, SVMs tend to behave better than logistic regression."

TUNING COST

11. Generate two-class data with classes barely linearly separable

```
q11_n_obs <- 500
q11_min <- -1
q11_max <- 1
q11_data <- tibble(X1 = runif(q11_n_obs, q11_min, q11_max),
               X2 = runif(q11_n_obs, q11_min, q11_max),
               perturb = rnorm(500, 0, 0.1))

q11_classify <- function(X1, X2, perturb) {
  if (X1 + X2 + perturb < 0)
    output <- "one"
  else
    output <- "two"
  return(output)
}

q11_data <- q11_data %>%
  rowwise() %>%
  mutate(class = q11_classify(X1, X2, perturb))

q11_data$class <- as.factor(q11_data$class)

q11_split <- initial_split(q11_data, prop = 0.7)
q11_train <- training(q11_split)

ggplot(q11_data, aes(x = X1, y = X2)) +
  geom_point(aes(color = class))
```



12. Compute the cross-validation error rates for SVCs with a range of cost values and number of training errors for each value of cost considered

```
q12_function <- function(cost) {

  svc_comp <- train(
  class ~ .,
  data = q11_train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneGrid = expand.grid(C = cost)
  )

  svc_pred <- svc_comp %>% predict(q11_train)
  svc_cm <- confusionMatrix(data = svc_pred, reference = q11_train$class)

  svc_error <- 1 - svc_cm$overall['Accuracy']
  svc_misclass <- as.data.frame(svc_cm$table) %>%
   filter(Prediction == "two" & Reference == "one" |
           Prediction == "one" & Reference == "two") %>%
   summarize(misclass = sum(Freq)) %>%
    as.numeric()

  svc_list <- list(error = svc_error, misclass = svc_misclass)
  return(svc_list)
}

cost <- 10^c(-5:10)
cost <- as.data.frame(cost)
q12_table <- cbind(cost, map_dfr(cost$cost, q12_function))
```

```
## maximum number of iterations reached 0.008761491 0.008693283
```

Relation to cross-validation errors obtained

```
q12_table
```

```
##     cost       error misclass
## 1  1e-05 0.214285714       75
## 2  1e-04 0.185714286       65
## 3  1e-03 0.194285714       68
## 4  1e-02 0.017142857        6
## 5  1e-01 0.011428571        4
## 6  1e+00 0.008571429        3
## 7  1e+01 0.005714286        2
## 8  1e+02 0.005714286        2
## 9  1e+03 0.011428571        4
## 10 1e+04 0.002857143        1
## 11 1e+05 0.000000000        0
## 12 1e+06 0.005714286        2
## 13 1e+07 0.005714286        2
## 14 1e+08 0.005714286        2
## 15 1e+09 0.005714286        2
## 16 1e+10 0.008571429        3
```

As the number of cross-validation error rates (`error`) decreases, the number of training errors (`misclass`) decreases then increases.

13. Generate data set

```
q11_test <- testing(q11_split)
```

Compute test errors corresponding to each of the cost values

```
q13_function <- function(cost) {

  svc_comp <- train(
  class ~ .,
  data = q11_train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneGrid = expand.grid(C = cost)
  )

  svc_pred <- svc_comp %>% predict(q11_test)
  svc_cm <- confusionMatrix(data = svc_pred, reference = q11_test$class)

  svc_error <- 1 - svc_cm$overall['Accuracy']
  svc_misclass <- as.data.frame(svc_cm$table) %>%
   filter(Prediction == "two" & Reference == "one" |
           Prediction == "one" & Reference == "two") %>%
   summarize(misclass = sum(Freq)) %>%
    as.numeric()

  svc_list <- list(error = svc_error, misclass = svc_misclass)
  return(svc_list)
}
```

Value of cost leading to fewest test errors

```
q13_table <- cbind(cost, map_dfr(cost$cost, q13_function))
```

```
## maximum number of iterations reached 0.008517305 0.008473723
```

```
q13_table
```

```
##     cost       error misclass
## 1  1e-05 0.206666667       31
## 2  1e-04 0.166666667       25
## 3  1e-03 0.166666667       25
## 4  1e-02 0.040000000        6
## 5  1e-01 0.020000000        3
## 6  1e+00 0.006666667        1
## 7  1e+01 0.006666667        1
## 8  1e+02 0.006666667        1
## 9  1e+03 0.006666667        1
## 10 1e+04 0.006666667        1
## 11 1e+05 0.000000000        0
## 12 1e+06 0.006666667        1
## 13 1e+07 0.006666667        1
## 14 1e+08 0.006666667        1
## 15 1e+09 0.006666667        1
## 16 1e+10 0.006666667        1
```

Comparison to values of cost leading to fewest training errors and fewest cross-validation errors

```
q13_comp <- tibble(type = c("training", "test"),
                   min_error = c(q12_table$error[which.min(q12_table$error)],
                             q13_table$error[which.min(q13_table$error)]),
                   error_cost = c(q12_table$cost[which.min(q12_table$error)],
                             q13_table$cost[which.min(q13_table$error)]),
                  min_misclass = c(q12_table$misclass[which.min(q12_table$misclass)],
                             q13_table$misclass[which.min(q13_table$misclass)]),
                   misclass_cost = c(q12_table$cost[which.min(q12_table$misclass)],
                             q13_table$cost[which.min(q13_table$misclass)]))

q13_comp
```

```
## # A tibble: 2 x 5
##   type     min_error error_cost min_misclass misclass_cost
##   <chr>        <dbl>      <dbl>        <dbl>         <dbl>
## 1 training         0     100000            0        100000
## 2 test             0     100000            0        100000
```

In the training data set, a cost value of 100,000 leads to a cross-validation error rate of 0 and 0 training errors. In the test data set, a cost value of 100,000 leads to a cross-validation error rate of 0 and 0 training errors.

14. Discussion

The above tables show that there is some basis to the claim that in the case of data that is just barely linearly separable, an SVC (Linear) with a small cost that misclassifies a couple of training observations may perform better than a SVC (Linear) with a huge cost that does not misclassify any data. Observe that the training and testing data quickly reach their minimum cross-validation error rates and misclassification error counts before increasing in error again. This may be a sign that at large costs, the linear kernel of the SVC overfits the data. It may thus be better to accept a few misclassifications and an overall lower error rate.

APPLICATION

15. Fit SVC to predict `colrac` as a function of all available predictors

```
train <- read.csv("data/gss_train.csv")
test <- read.csv("data/gss_test.csv")
train$colrac <- as.factor(train$colrac)
test$colrac <- as.factor(test$colrac)
train <- train  %>%
  mutate(colrac = factor(colrac,
          labels = make.names(levels(colrac))))
test <- test  %>%
  mutate(colrac = factor(colrac,
          labels = make.names(levels(colrac))))
```

Use 10-fold CV to find optimal value for `cost`

```
gss_svc <- train(
  colrac ~ .,
  data = train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneGrid = expand.grid(C = 10^c(2))
  )
```

Report CV errors associated with different values of `cost`

```
q15_table <- tibble(cost = gss_svc$results$C,
                    cv_err = 1 - gss_svc$results$Accuracy)
q15_table
```

```
## # A tibble: 8 x 2
##        cost cv_err
##       <dbl>  <dbl>
## 1   0.00001  0.303
## 2   0.0001   0.302
## 3   0.001    0.215
## 4   0.01     0.199
## 5   0.1      0.206
## 6   1        0.206
## 7  10        0.211
## 8 100        0.212
```

Discuss results

As the cost increases, the CV error decreases to a minimum of 0.1985238 at a cost of 0.01, and then increases. This may be a sign that the model is overfitting the data.

16. Fit SVMs with radial and polynomial kernels

Use 10-fold CV to find optimal value for `gamma`, `degree`, and `cost`

```
gss_radial <- train(
  colrac ~ .,
  data = train,
  method = "svmRadial",
  trControl = cv_ctrl,
  tuneGrid = expand.grid(C = 10^c(-5:0),
                         sigma = 10^c(-5:0))
  )

gss_poly <- train(
  colrac ~ .,
  data = train,
  method = "svmPoly",
  trControl = cv_ctrl,
  tuneGrid = expand.grid(C = 10^c(-5:0),
                         scale = 0.1,
                         degree = 1:5)
  )
```

Present results

```
q16_table_r <- tibble(cost = gss_radial$results$C,
                      gamma = gss_radial$results$sigma,
                      cv_err = 1 - gss_radial$results$Accuracy)  %>%
  arrange(cv_err)

q16_table_p <- tibble(cost = gss_poly$results$C,
                      degree = gss_poly$results$degree,
                      cv_err = 1 - gss_poly$results$Accuracy) %>%
  arrange(cv_err)

q16_table_r
q16_table_p
```

```
## # A tibble: 36 x 3
##       cost gamma cv_err
##      <dbl> <dbl>  <dbl>
##   1 1       0.01   0.196
##   2 1       0.001  0.212
##   3 0.1     0.01   0.221
##   4 0.01    0.1    0.255
##   5 0.1     0.1    0.255
##   6 1       0.1    0.257
##   7 0.00001 0.01   0.280
##   8 0.001   0.1    0.284
##   9 0.001   0.01   0.295
## 10 0.01    0.01   0.296
## # ... with 26 more rows
```

```
## # A tibble: 30 x 3
##      cost degree cv_err
##     <dbl>  <int>  <dbl>
##  1 0.01        2  0.198
##  2 0.001       4  0.201
##  3 0.01        3  0.202
##  4 0.1         2  0.206
##  5 0.1         1  0.206
##  6 0.01        4  0.207
##  7 0.1         4  0.208
##  8 1           4  0.209
##  9 0.0001      5  0.209
## 10 0.001       5  0.209
## # ... with 20 more rows
```

Discuss results

In the case of the SVM (Radial), the minimum CV error of 0.1958388 occurs at a cost of 1 and a gamma of 0.01. In the case of the SVM (Polynomial), the minimum CV error of 0.1978430 occurs at a cost of 0.01 and a degree of 2. The CV error results of the SVM (Radial) and the SVM (Polynomial) are highly similar to the CV error result of the SVC (Linear) (0.1985238). This suggests that the models can yield similar fits, albeit at different costs and using different hyperparameters.