Problem Set 6

Pete Cuppernull

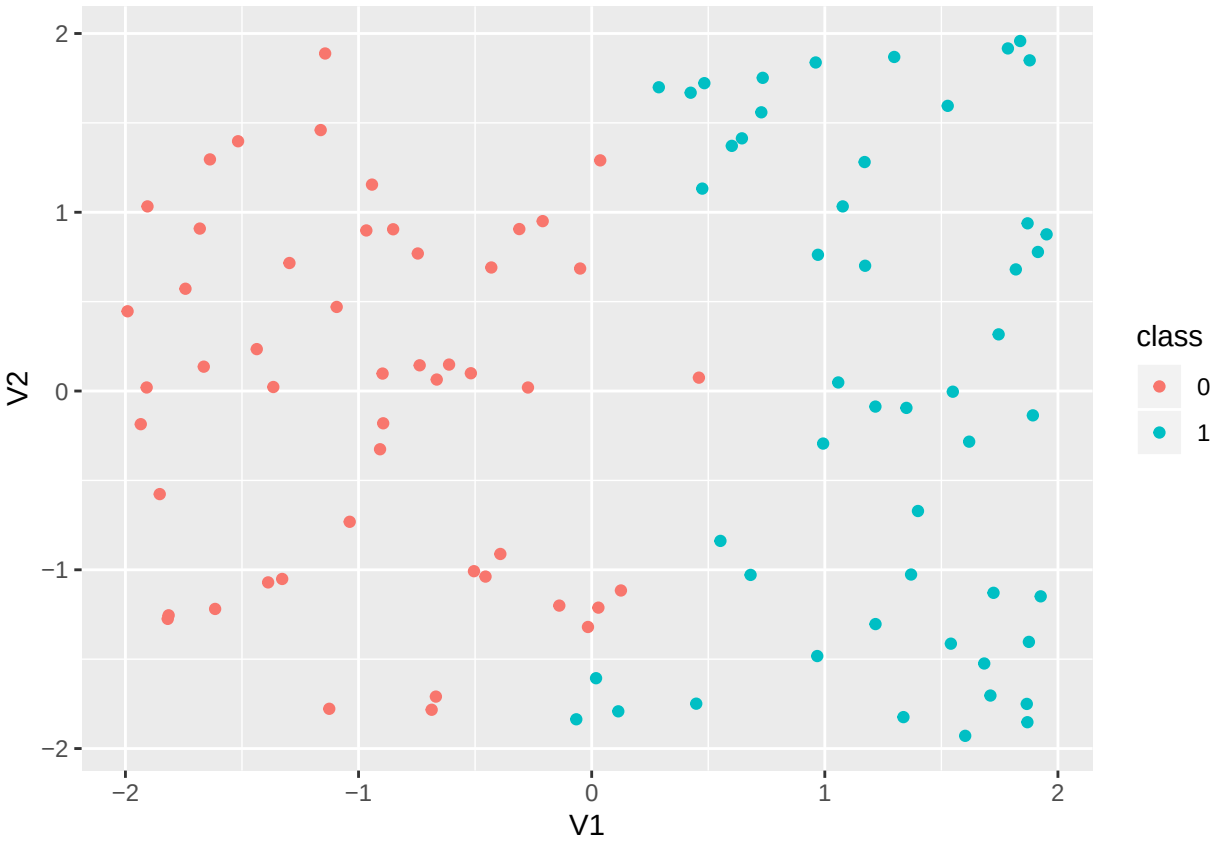
3/8/2020

Conceptual Exercises

1. Non-linear Separation

Create Data

```
set.seed(1414)

#training data
test <- as.data.frame(matrix(runif(200, -2, 2), ncol = 2))
test <- test %>%
  mutate(order = V2^2 + V1*2) %>%
  arrange(order)
class <- c(rep(0, 50), rep(1, 50))

concept <- cbind(test, class) %>%
  mutate(class = as.factor(class))

#test data
test2 <- as.data.frame(matrix(runif(200, -2, 2), ncol = 2))
test2 <- test2 %>%
  mutate(order = V2^2 + V1*2) %>%
  arrange(order)

concept_test <- cbind(test2, class) %>%
  mutate(class = as.factor(class))


ggplot(concept) +
  geom_point(mapping = aes(V1, V2, color = class))
```



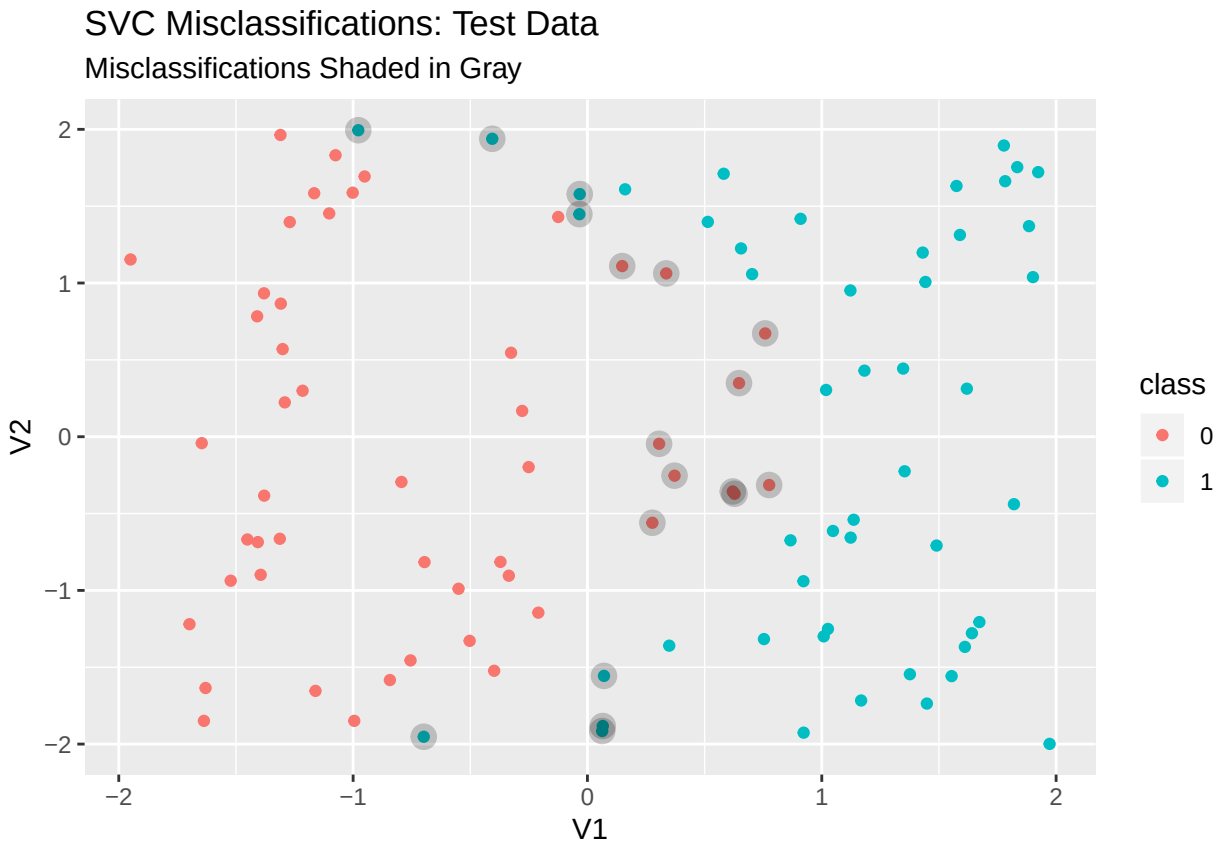
Fit Models

```
#Linear
concept_lin <- svm(
        class ~ V1 + V2,
        data = concept,
        type = "C-classification",
        kernel = "linear")

#Training error
concept_lin_error <- cbind(concept, concept_lin$fitted) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`concept_lin$fitted`))) %>%
  summarize(accuracy = sum(accuracy)/n())

#Test error
lin_preds <- predict(concept_lin, concept_test)

concept_lin_error_test <- cbind(concept_test, lin_preds) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`lin_preds`))) %>%
  summarize(accuracy = sum(accuracy)/n())


#Radial
concept_radial <- svm(
        class ~ V1 + V2,
        data = concept,
        type = "C-classification",
        kernel = "radial")
```

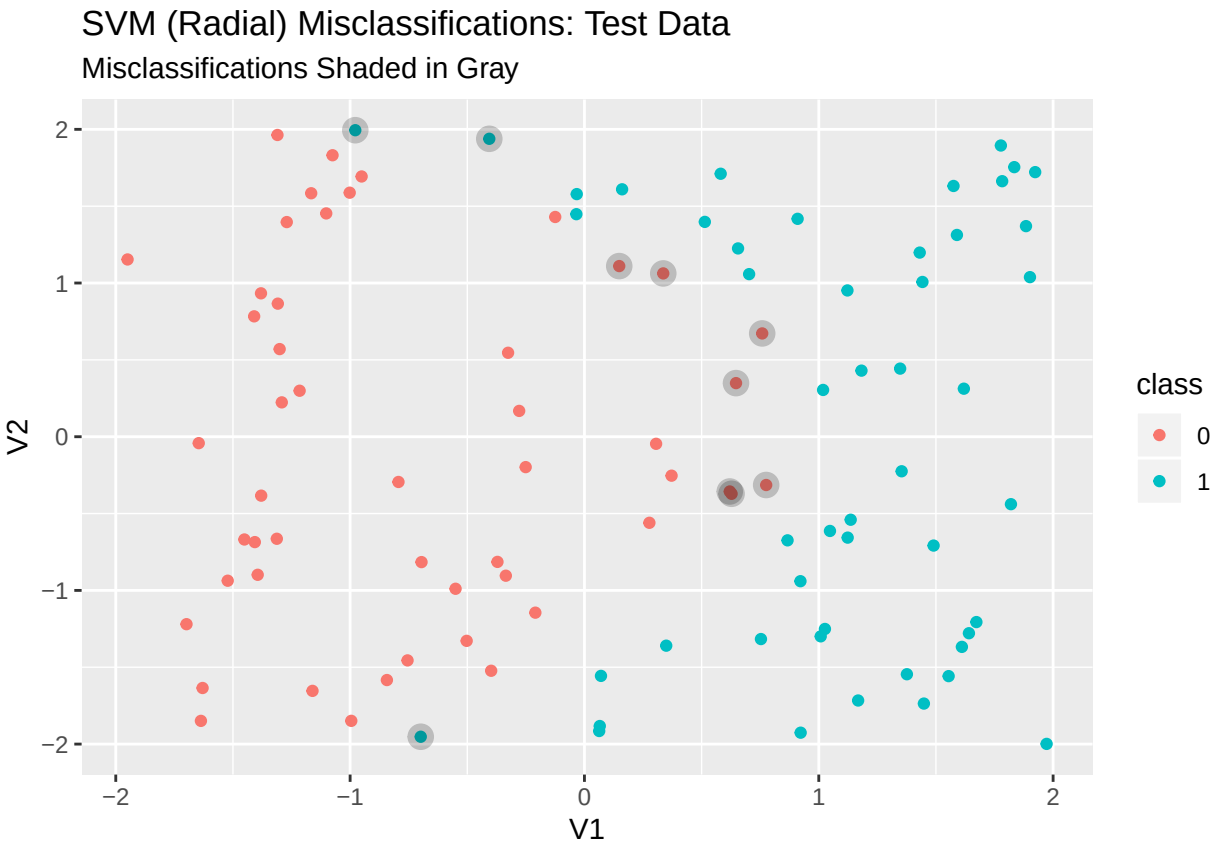
```r
#Training Error
concept_radial_error <- cbind(concept, concept_radial$fitted) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`concept_radial$fitted`))) %>%
  summarize(accuracy = sum(accuracy)/n())

#Test Error
radial_preds <- predict(concept_radial, concept_test)

concept_radial_error_test <- cbind(concept_test, radial_preds) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`radial_preds`))) %>%
  summarize(accuracy = sum(accuracy)/n())
```

The support vector classifer training error is 0.04 and the support vector machine training error with a radial kernel is 0. As for the test data, the SVC test error is 0.18 and the SVM test error with the radial kernal is 0.1. As expected, the radial technique performs better for both the training and testing data sets – the original data classifications were generated by a non-linear function, and so we can expect that a nonlinear model would perform best (which it did). Below, we can visually examine the misclassifications prduced by each model.

```r
#Plot for Linear
lin_misclass <- cbind(concept_test, lin_preds) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`lin_preds`))) %>%
  filter(accuracy == 1)

lin_misclass_all <- cbind(concept_test, lin_preds) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`lin_preds`)))

ggplot() +
  geom_point(data = lin_misclass_all, mapping = aes(x = V1, y = V2, color = class)) +
  geom_point(data = lin_misclass, mapping = aes(x = V1, y = V2), alpha = .2, size = 4) +
  labs(title = "SVC Misclassifications: Test Data",
       subtitle = "Misclassifications Shaded in Gray")
```




```
#Plot for Radial

radial_misclass <- cbind(concept_test, radial_preds) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`radial_preds`))) %>%
  filter(accuracy == 1)

radial_misclass_all <- cbind(concept_test, radial_preds) %>%
  mutate(accuracy = abs(as.numeric(class) - as.numeric(`radial_preds`)))

ggplot() +
  geom_point(data = radial_misclass_all, mapping = aes(x = V1, y = V2, color = class)) +
  geom_point(data = radial_misclass, mapping = aes(x = V1, y = V2), alpha = .2, size = 4) +
  labs(title = "SVM (Radial) Misclassifications: Test Data",
       subtitle = "Misclassifications Shaded in Gray")
```



SVM vs. Logistic Regression

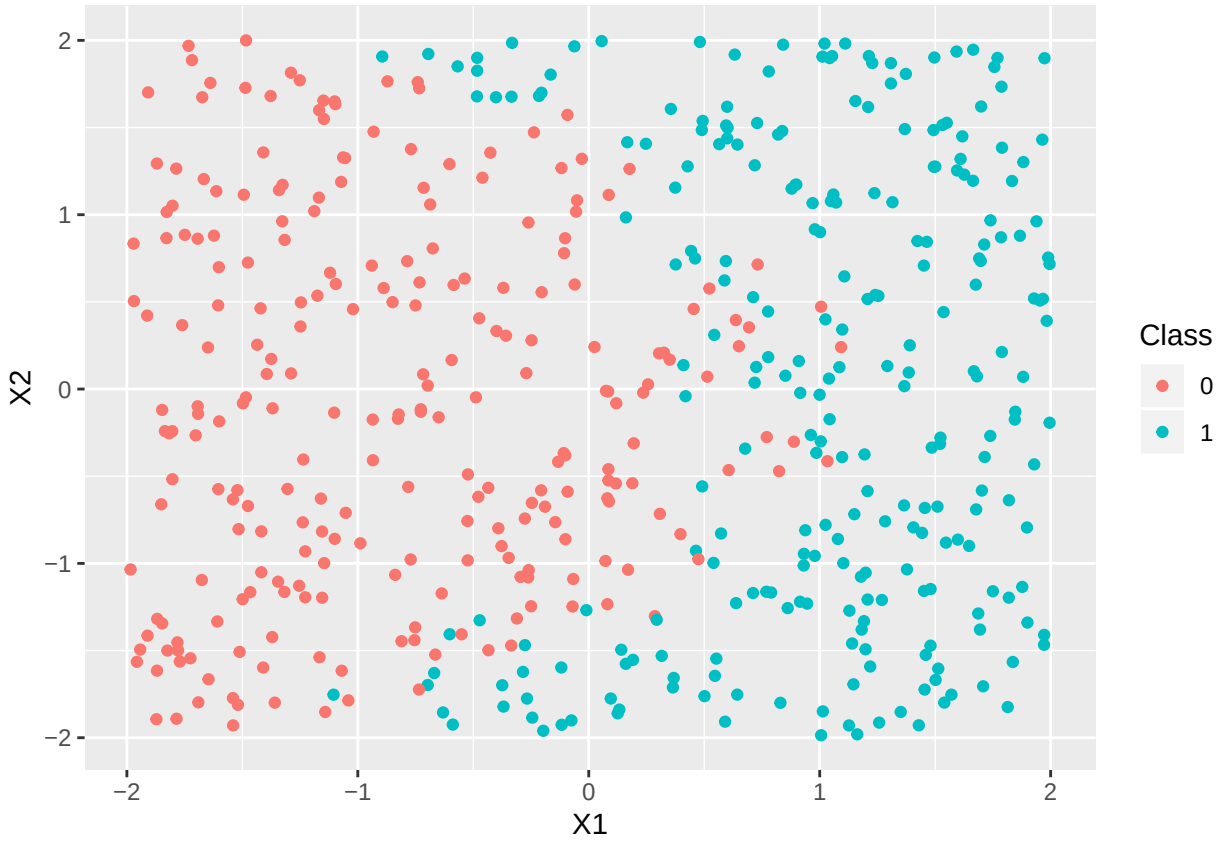
2. Create data

```
test2 <- as.data.frame(matrix(runif(1000, -2, 2), ncol = 2))

test2 <- test2 %>%
  mutate(order = V2^2 + V1*2) %>%
  arrange(order)
class2 <- c(rep(0, 200), rep(1, 5), rep(0, 5), rep(0, 5), rep(0, 5),
            rep(1, 5), rep(0, 5), rep(0, 5), rep(1, 5), rep(0, 5),
            rep(1, 5), rep(0, 5), rep(1, 5), rep(1, 5), rep(0, 5),
            rep(1, 5), rep(1, 5), rep(1, 5), rep(0, 5), rep(1, 210))

test2 <- cbind(test2, class2)
```

3. Plot Data

```
test2 <- test2 %>%
  rename(X1 = V1,
         X2 = V2) %>%
  mutate(Class = as.factor(class2))
```

```r
ggplot(test2) +
  geom_point(mapping = aes(X1, X2, color = Class))
```

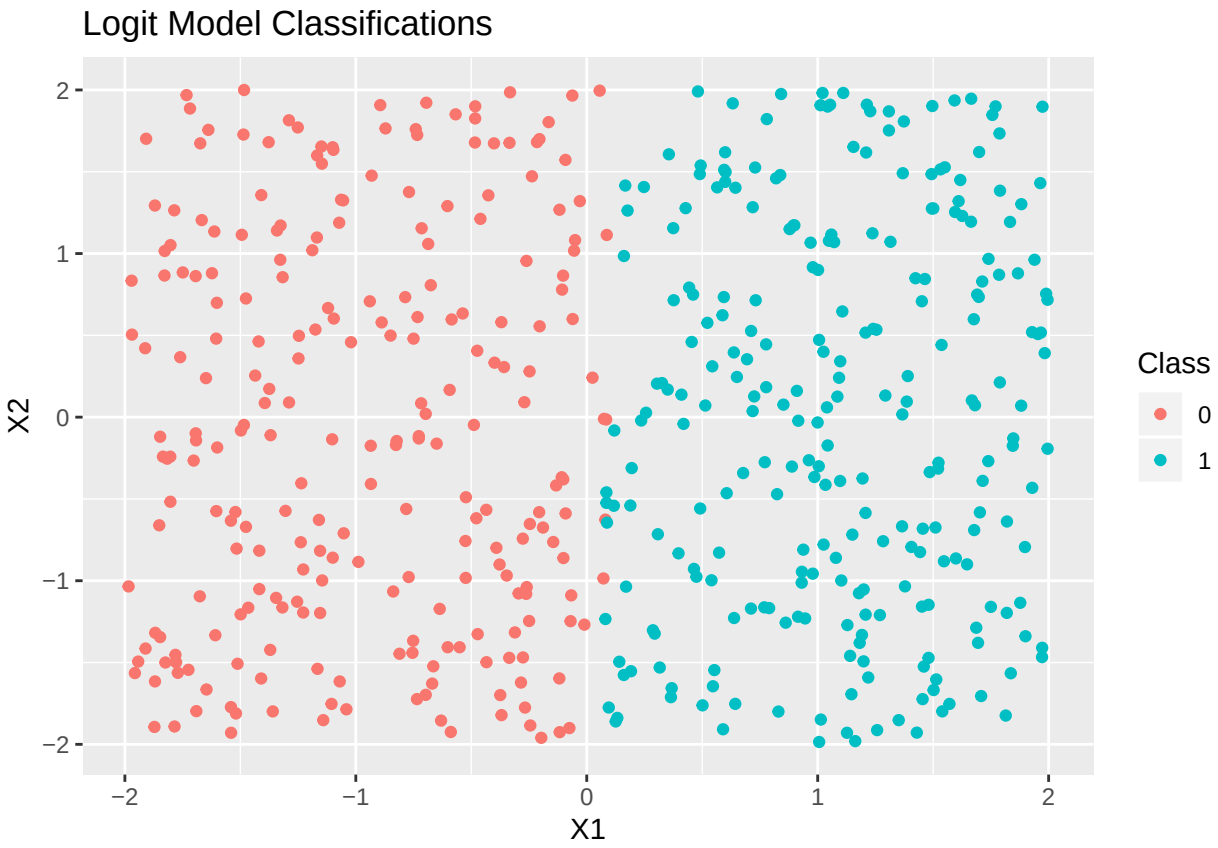


4. Fit Logit Model

```r
logit_mod <- glm(Class ~ X1 + X2, family = "binomial", data = test2)
```

5. Obtain Predictions and Plot

```r
predictors <- test2[,1:2]
preds <- predict(logit_mod,
        newx = predictors)

logit_pred_data <- cbind(predictors,preds) %>%
  mutate(Class = as.factor(if_else(preds < 0, 0, 1)))

ggplot(logit_pred_data) +
  geom_point(mapping = aes(X1, X2, color = Class)) +
  labs(title = "Logit Model Classifications")
```

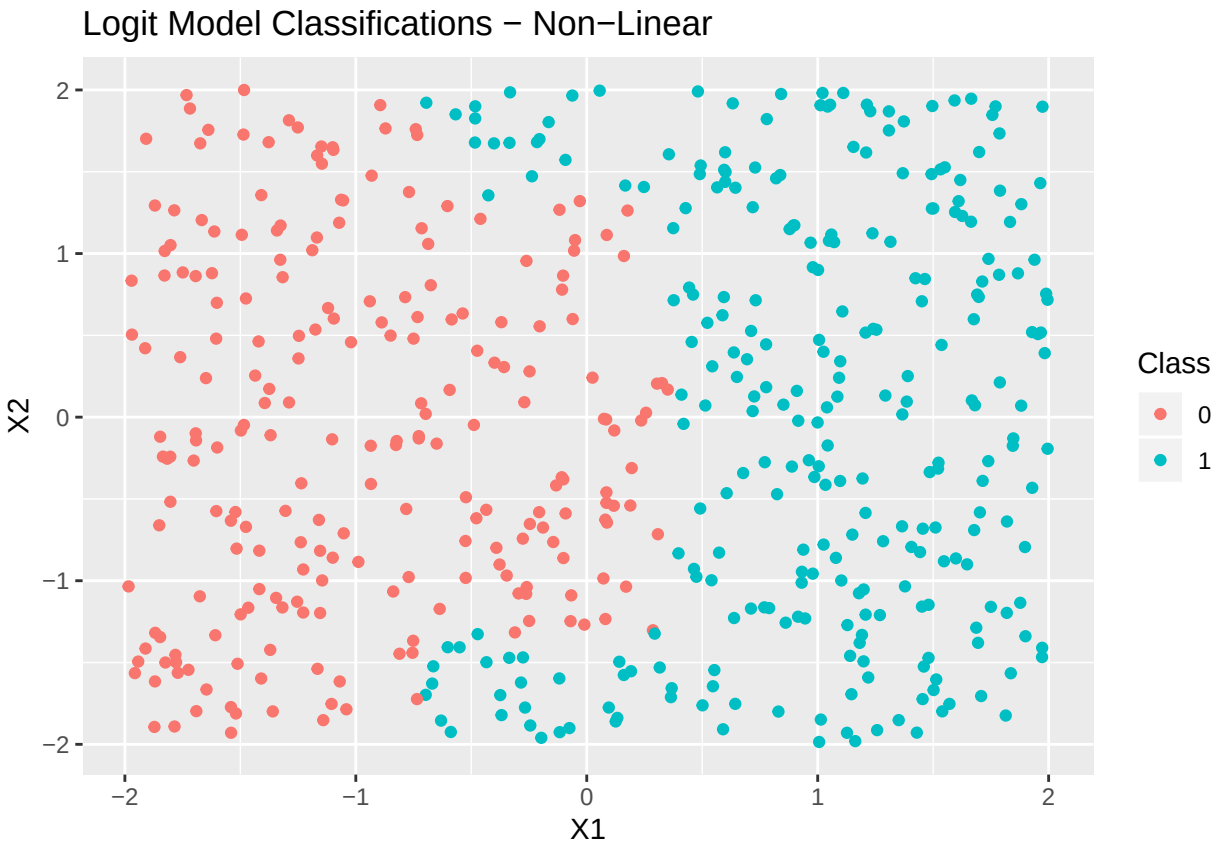


6. Fit non-linear logit

```
logit_mod_NL <- rpart(Class ~ X1 + poly(X2, 2), data = test2)
```

7. Obtain class labels and plot

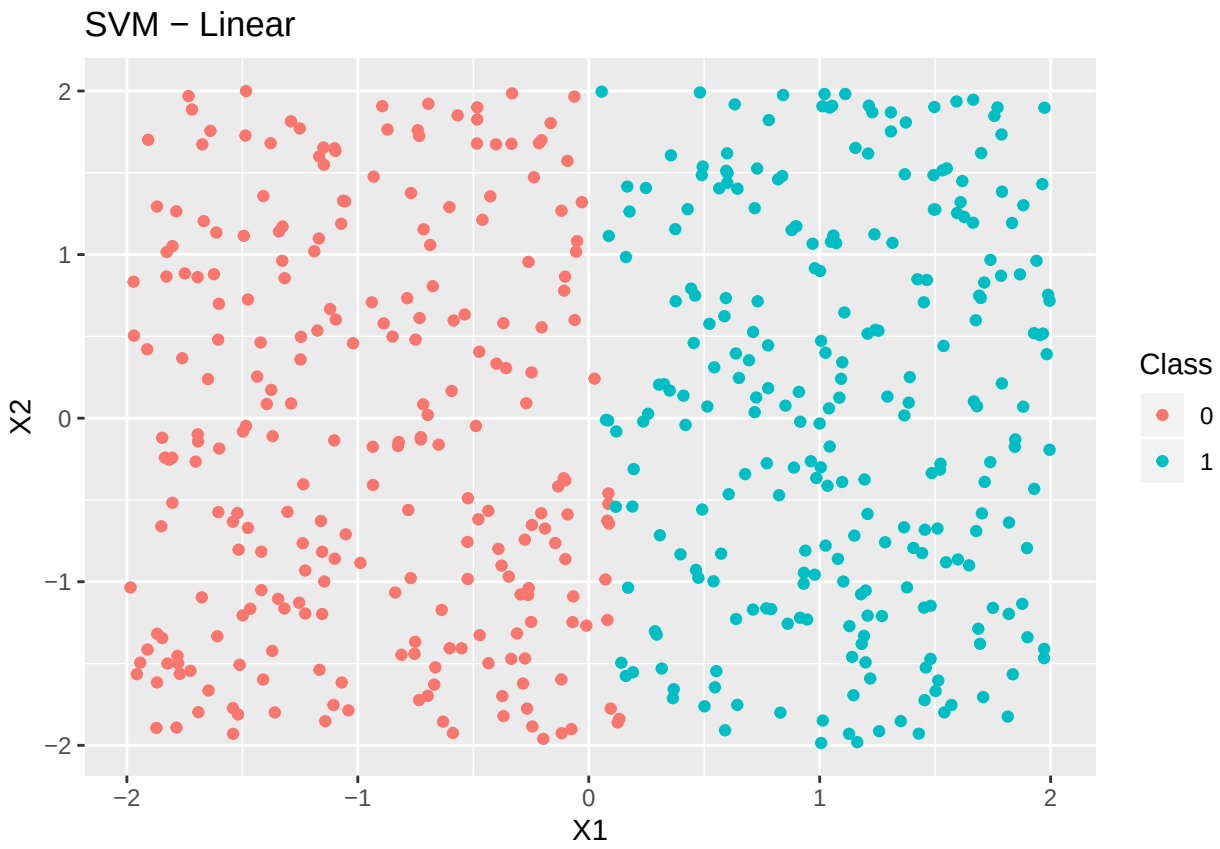
```
preds_NL <- predict(logit_mod_NL,
        newx = predictors,
        type = "class")


logit_pred_data_NL2 <- cbind(predictors,preds_NL) %>%
  mutate(Class = preds_NL)

ggplot(logit_pred_data_NL2) +
  geom_point(mapping = aes(X1, X2, color = Class)) +
  labs(title = "Logit Model Classifications - Non-Linear")
```



8. Fit SVM with linear kernel

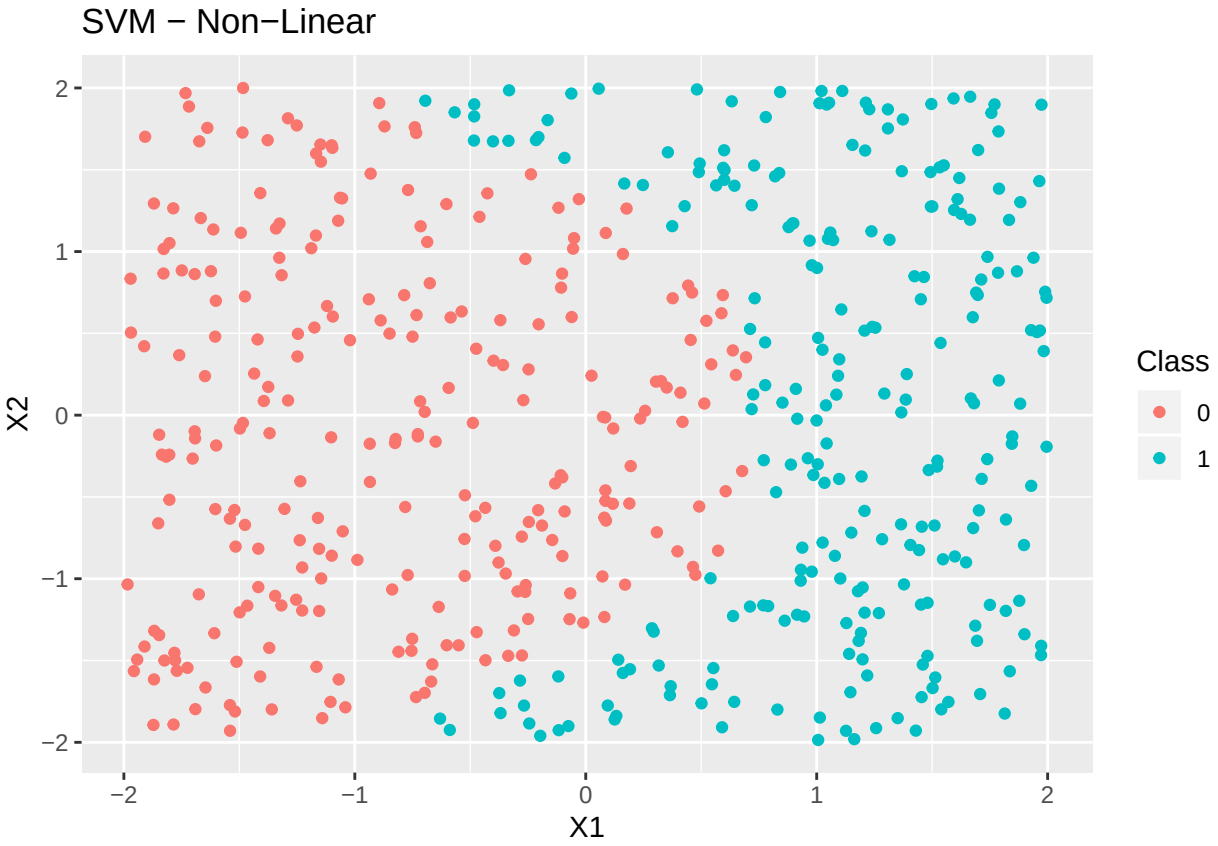
```
test3 <- test2 %>%
  select(X1, X2, Class) %>%
  mutate(Class = as.factor(if_else(Class == 0, "No", "Yes")))


cv_ctrl <- trainControl(savePredictions = "final",
                        classProbs = TRUE)

svm_linear <- train(
  Class ~ X1 + X2,
  data = test3,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10,
  family = "binomial"
)


preds_svm_lin <- predict(svm_linear,
        newdata = predictors)

svm_pred_data_lin <- cbind(predictors,preds_svm_lin) %>%
  mutate(Class = as.factor(if_else(preds_svm_lin == "No", 0, 1)))
```

```
ggplot(svm_pred_data_lin) +
  geom_point(mapping = aes(X1, X2, color = Class)) +
  labs(title = "SVM - Linear")
```



9. SVM Non-linear

```
svm_poly <-
    svm(Class ~ X1 + poly(X2, 2),
        data = test3,
        type = "C-classification",
        kernel = "polynomial",
        degree = 3)



preds_svm_poly <- predict(svm_poly,
        newdata = predictors)

svm_pred_data_poly <- cbind(predictors,preds_svm_poly) %>%
  mutate(Class = as.factor(if_else(preds_svm_poly == "No", 0, 1)))

ggplot(svm_pred_data_poly) +
  geom_point(mapping = aes(X1, X2, color = Class)) +
  labs(title = "SVM - Non-Linear")
```



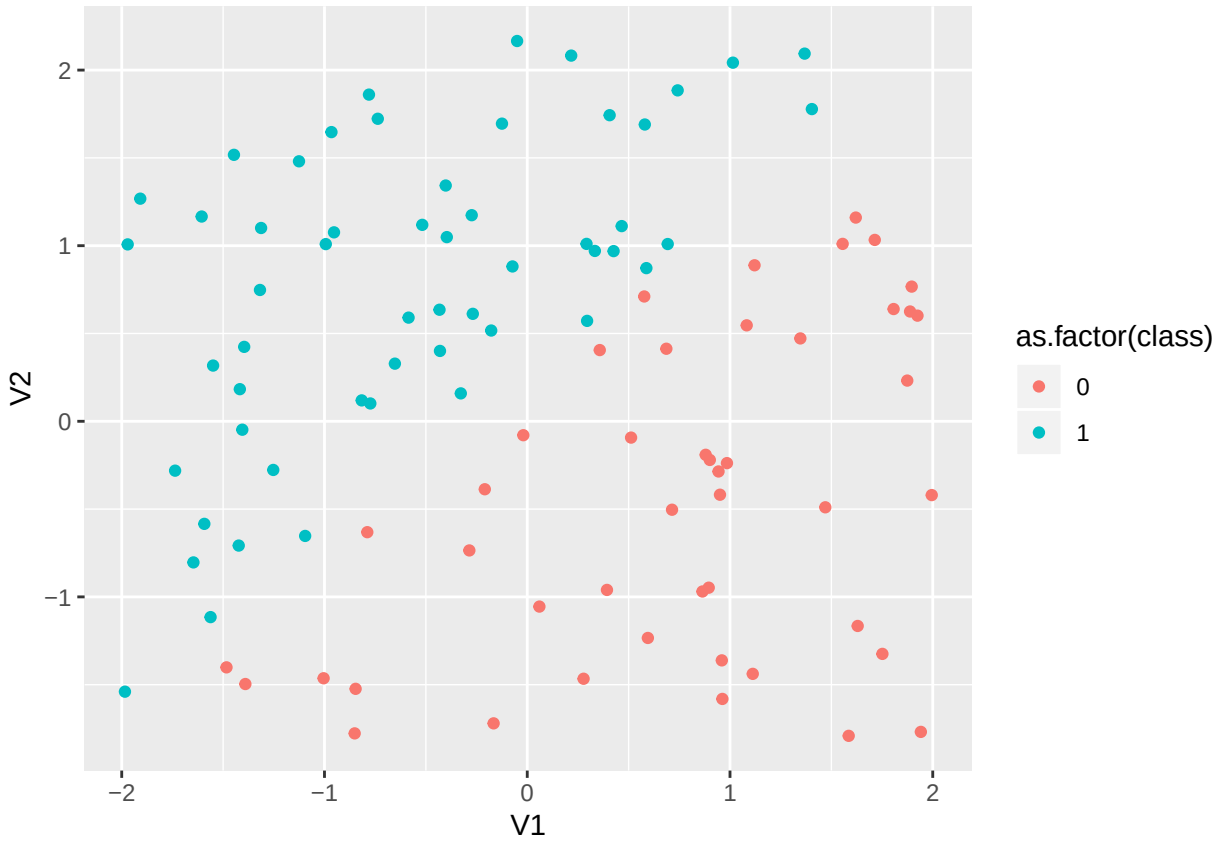
10. Discussion

As we can see, both of the non-linear methods outperform the linear methods for modeling the data - this is the expected result, considering that the data was created with a non-linear decision boundary. For this example, logistic regression would probably be favorable because the data was created without any outliers – however, considering that in the real world we will likely not be certain of the data generating process and thus need to be weary of outliers, a SVM would be favorable because it is less sensitive to outliers.

Tuning Cost

11. Create New Data

```
tune_test <- as.data.frame(matrix(runif(200, -2, 2), ncol = 2))

tune_test2 <- tune_test %>%
  mutate(class = if_else(V2-V1 > 0, 1, 0),
         V2 = V2 + .2)


ggplot(tune_test2) +
  geom_point(mapping = aes(V1, V2, color = as.factor(class)))
```



12. SVCs with a range of cost

```
tune_test2 <- tune_test2 %>%
  mutate(class = as.factor(if_else(class == 0, "no", "yes")))

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


cv_ctrl_cost2 <- trainControl(method = "cv",
                         number = 10)


svm_linear_cost2 <- train(
  class ~ V1 + V2,
  data = tune_test2,
  method = "svmLinear",
  trControl = cv_ctrl_cost2,
  tuneGrid = expand.grid(C = c(.0001, .001, .01, .1, 1, 10, 100)),
  tuneLength = 10
)
```

```
cost_tests <- svm_linear_cost2$results

cost_tests %>%
  select(C, Accuracy) %>%
  mutate(n_misclass = round(100*(1 - Accuracy)))
```

```
##       C  Accuracy n_misclass
## 1 1e-04 0.5403030         46
## 2 1e-03 0.5403030         46
## 3 1e-02 0.8573737         14
## 4 1e-01 0.9386869          6
## 5 1e+00 0.9800000          2
## 6 1e+01 0.9900000          1
## 7 1e+02 1.0000000          0
```

```
cost_tests %>%
  select(C, Accuracy) %>%
  mutate(n_misclass = round(100*(1 - Accuracy))) %>%
  ggplot(aes(x = log(C), y = n_misclass)) +
  geom_line() +
  labs(title = "Number of Training Misclassifications across Range of Cost",
       subtitle = "Out of 100 Observations")
```

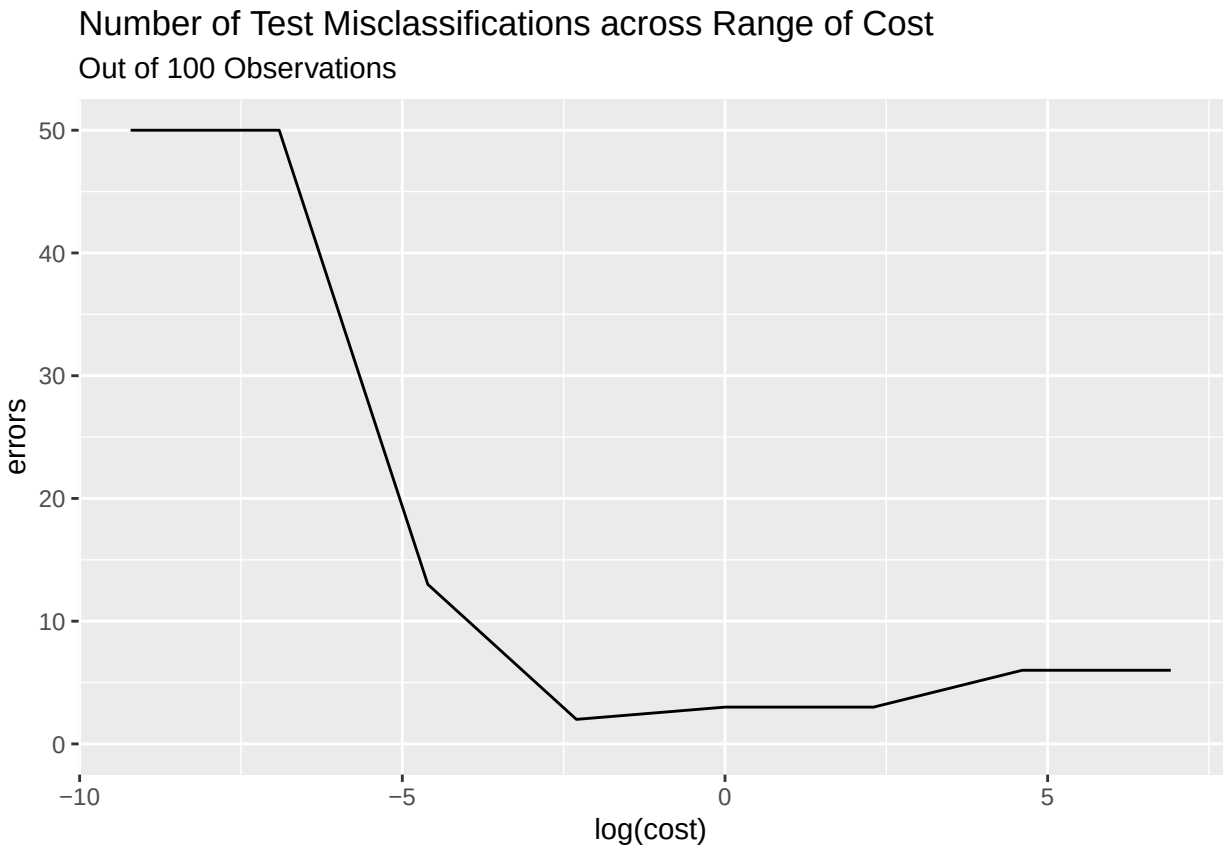
The number of traning classification errors is inversely related to the cost - as the cost increases, the number of misclassifications decreases.

13. Generate Test Data and Compute Errors

```
set.seed(2)
test_data <- as.data.frame(matrix(runif(200, -2, 2), ncol = 2))

test_data <- test_data %>%
  mutate(class = if_else(V2-V1 > 0, 1, 0))


test_function2 <- function(cost){
  svm <- svm(class ~ V1 + V2,
        data = tune_test2,
        type = "C-classification",
        kernel = "linear",
        cost = cost)


  preds <- predict(svm,
        newdata = test_data[,-3])

  errors <- cbind(test_data, preds) %>%
    mutate(preds = if_else(preds == "yes", 1, 0)) %>%
    mutate(accuracy = if_else(class == preds, 1, 0)) %>%
    summarize(errors = sum(1- accuracy))

 errors[1,1]

}


test_errors <- as.data.frame(cbind(c(.0001, .001, .01, .1, 1, 10, 100, 1000), map_dbl(c(.0001, .001, .0
  rename(cost = V1,
         errors = V2)



test_errors
```

```
##    cost errors
## 1 1e-04     50
## 2 1e-03     50
## 3 1e-02     13
## 4 1e-01      2
## 5 1e+00      3
## 6 1e+01      3
## 7 1e+02      6
## 8 1e+03      6
```

```
ggplot(test_errors, aes(x = log(cost), y = errors)) +
  geom_line() +
  labs(title = "Number of Test Misclassifications across Range of Cost",
       subtitle = "Out of 100 Observations") +
  ylim(0, 50)
```



13. Best Value of Cost?

The value of cost that led to the lowest testing error rate was .1, which produced 2 errors (out of 100 observations). This is different than the cost which produced the best training error rate, which was 1000.

14. Discussion

The reason different values of cost produced the best training and test error rates is because the high values of cost overfit the training data, producing low training error rates with models that did not generalize well to new data. The "U-shaped" curve of test error rates across values of cost indicated that that is an ideal middle point of cost between sufficient specificity and ample generalizability for the model, a nod to the more general notion of the bias/variance tradeoff.

Application

15. Fit SVC

```
gss_test <- na.omit(read_csv("data/gss_test.csv"))  %>%
  mutate(colrac = factor(colrac,
                         levels = c(0, 1),
                         labels = c("No", "Yes")))

gss_train <- na.omit(read_csv("data/gss_train.csv"))  %>%
  mutate(colrac = factor(colrac,
```

```
                          levels = c(0, 1),
                          labels = c("No", "Yes")))


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

grid <- expand.grid(C = c(.0001, .001, .01, .1, 1, 10))

# fit model

gss_linear <- train(
  colrac ~ .,
  data = gss_train,
  method = "svmLinear",
  trControl = app_ctrl,
    tuneGrid = grid,
  tuneLength = 10
)

gss_linear$results %>%
  select(C, Accuracy) %>%
  rename(Cost = C)
```

```
##    Cost  Accuracy
## 1 1e-04 0.6975124
## 2 1e-03 0.7812154
## 3 1e-02 0.7981165
## 4 1e-01 0.7866389
## 5 1e+00 0.7852829
## 6 1e+01 0.7873236
```

```
gss_linear$results$C[which.max(gss_linear$results$Accuracy)]
```

```
## [1] 0.01
```

The model with the highest accuracy is the one with a cost function of 0.01. As expected, the lowest and highest cost functions generally had poorer test accuracy because they were underfit and overfit to the training data, respectively.

16. SVM of radial and polynomial kernels

```
grid <- expand.grid(C = c(.0001, .001, .01, .1, 1, 10))

# fit models

#Poly
svm_poly <- svm(
        colrac ~ .,
        data = gss_train,
        type = "C-classification",
        kernel = "polynomial",
```

```
        degree = 3,
        gamma = .1,
        cost = .1,
        cross = 10)

mean(svm_poly$accuracies)
```

```
## [1] 77.37802
```

```
#write function
svm_search <- function(gamma, degree, cost){

  svm <- svm(
        colrac ~ .,
        data = gss_train,
        type = "C-classification",
        kernel = "polynomial",
        degree = degree,
        gamma = gamma,
        cost = cost,
        cross = 10)

mean(svm$accuracies)

}

svm_search_grid <- expand_grid(
  gamma = c(.001, .01, .1, 1, 10, 100),
  degree = seq(3, 6, 1),
  cost = c(.001, .01, .1, 1, 10, 100)
)

svm_search_results <- pmap_dbl(svm_search_grid, svm_search)

svm_poly <- cbind(svm_search_grid, svm_search_results) %>%
  rename(accuracy = svm_search_results) %>%
  arrange(desc(accuracy)) %>%
  head(5)

svm_poly
```

```
##   gamma degree  cost accuracy
## 1 1e-01      3 1e-02 80.42037
## 2 1e-02      3 1e+01 80.21767
## 3 1e-02      5 1e+02 79.67849
## 4 1e+00      3 1e-03 79.00190
## 5 1e+02      3 1e+00 78.93252
```

The best model has a gamma value of 0.1, a degree of 3, and a cost of 0.01, with a final accuracy of 80.42037. This would indicate that the learner can correctly predict the responses of 80.4% of respondents, a considerable improvement on a the naive accuracy rate of 53%. Across the best models, a polynomial fit of degree 3 tended to be the best, and as we highlighted earlier in the assignment, moderate cost values are preferable.