Chen_Zhaoyang HW6

Zhaoyang Chen

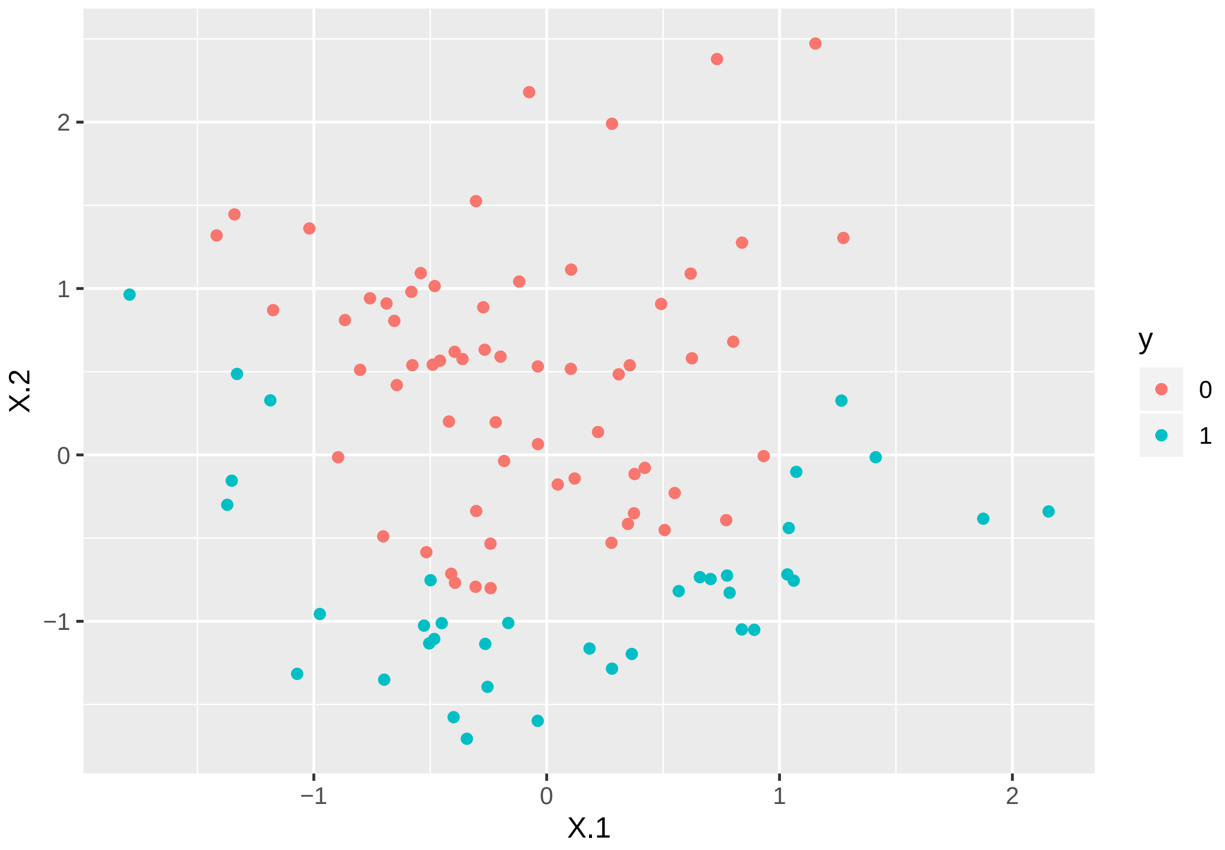
3/8/2020

```r
library(tidyverse)
library(caret)
library(e1071)
library(kernlab)
library(pROC)
```

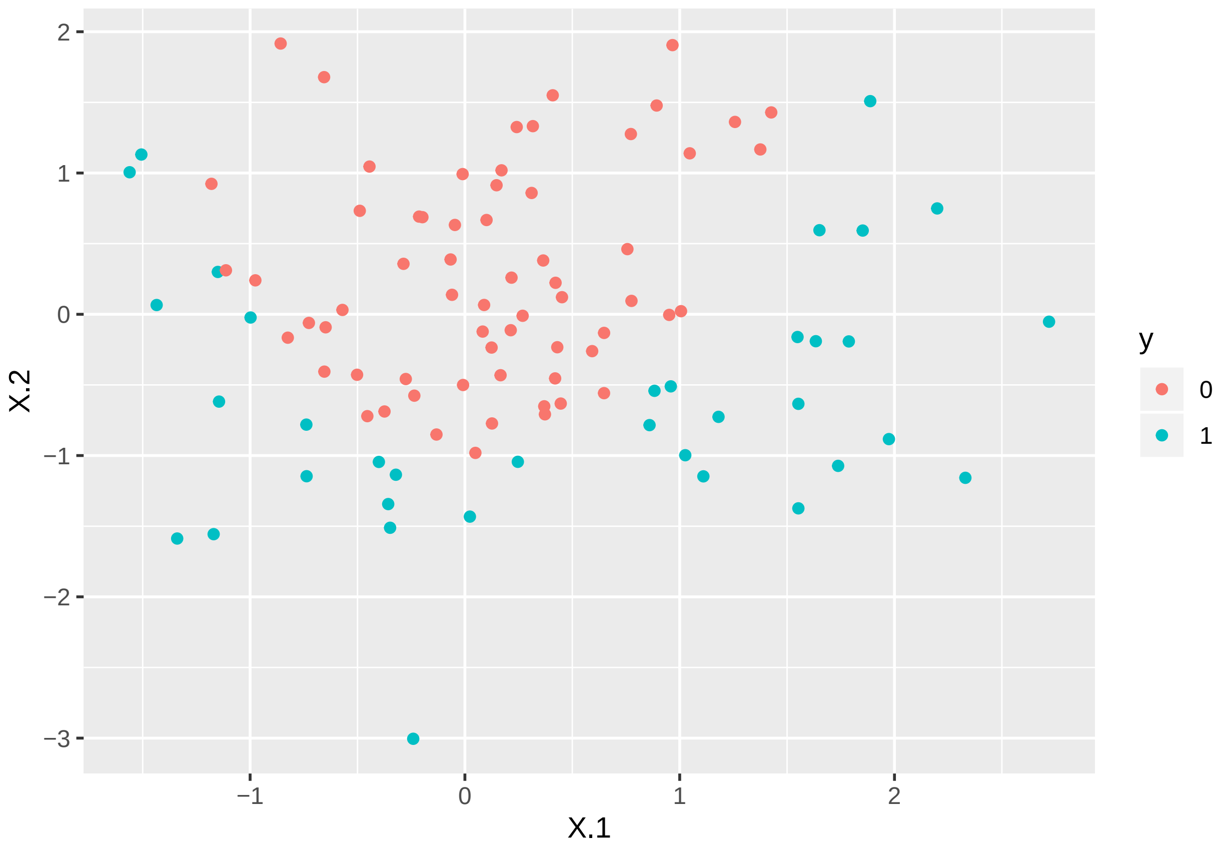
Conceptual Exercises

Non-linear separation

```r
x1 = rnorm(100)
x2 = rnorm(100)
y = (x2 <= x1^2 - 1)
train = tibble(X.1 = x1, X.2 = x2, y = factor(as.numeric(y)))
train %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y))
```



```r
x1 = rnorm(100)
x2 = rnorm(100)
y = (x2 <= x1^2 - 1)
test = tibble(X.1 = x1, X.2 = x2, y = factor(as.numeric(y)))
test %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y))
```



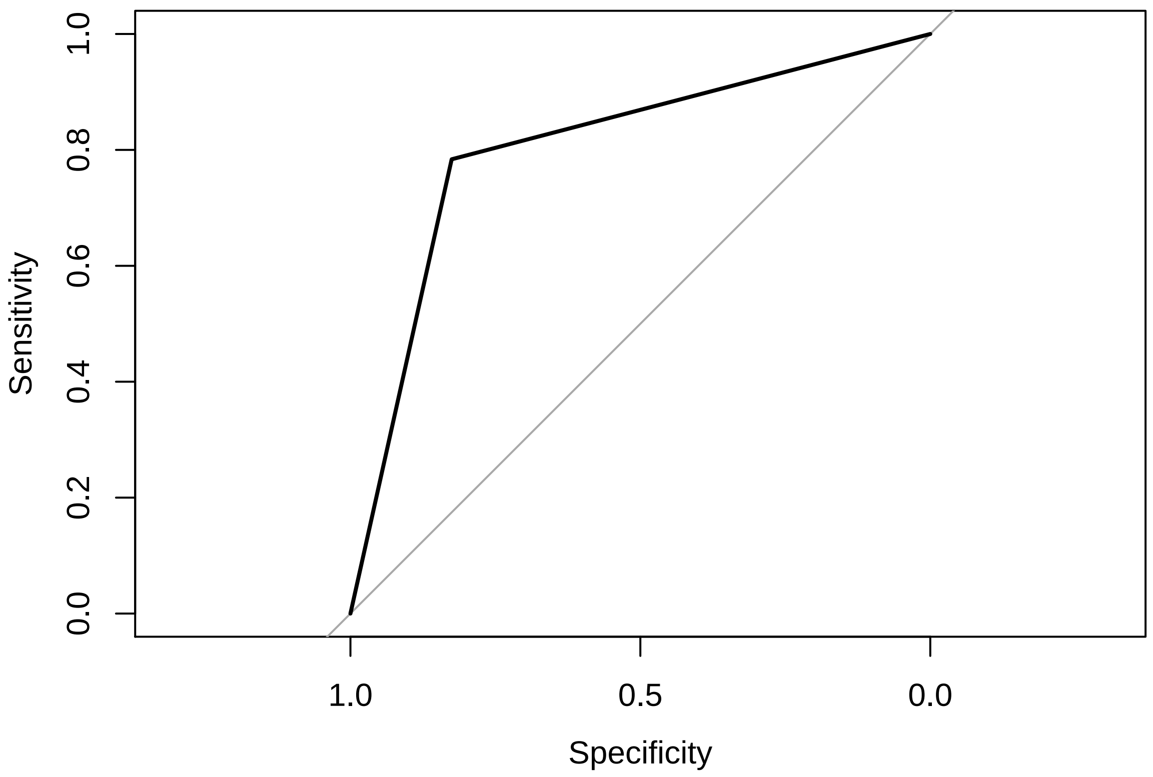
First, I generate the stimulated data with a quadratic function $f(x) = x^2 - 1$ and the result clearly shows a nonlinear relationship.

```
cv_ctrl = trainControl(method = "cv",
                        number = 10,
                        savePredictions = "final",
                        classProbs =F)

svm_linear = train(
  y ~ .,
  data = train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10
)

svm_linear_roc = roc(predictor = as.integer(svm_linear$pred$pred),
                      response = svm_linear$pred$obs,
                      levels = c(0,1))

plot(svm_linear_roc)
```



```
auc(svm_linear_roc)
```

```
## Area under the curve: 0.8046
```

```
cat('Error Rate is ')
```

```
## Error Rate is
```

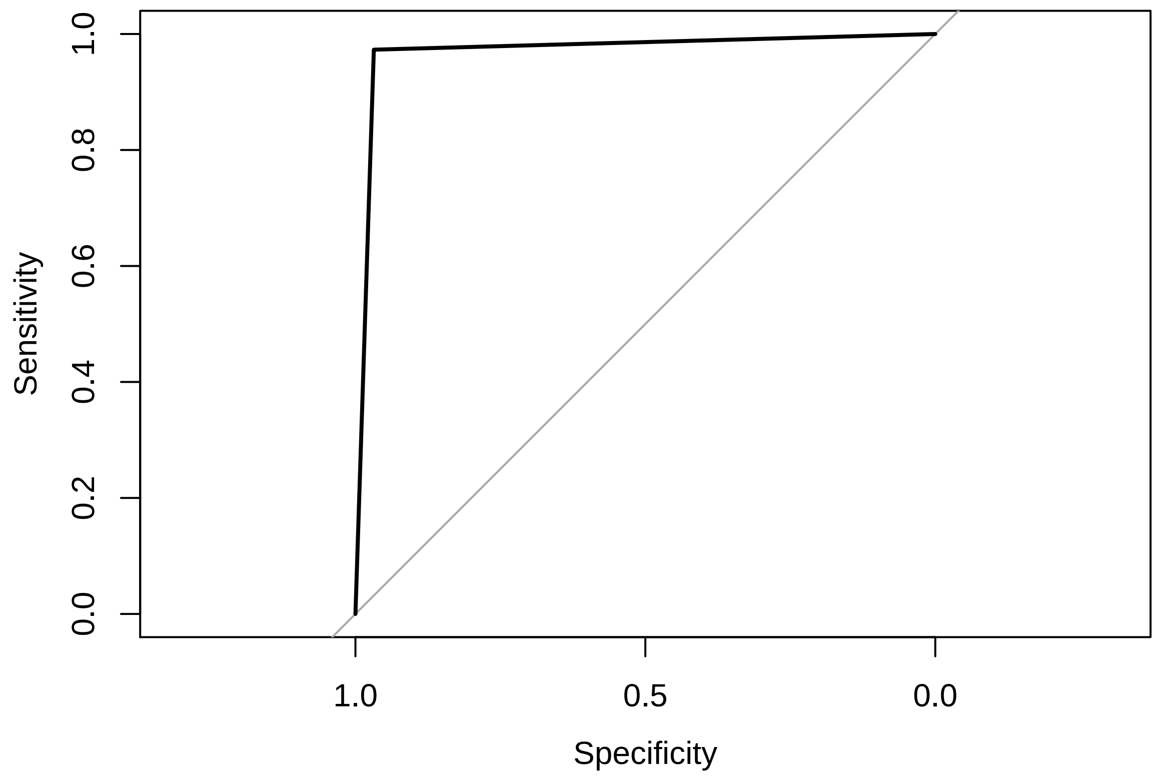
```
mean(svm_linear$pred$pred != svm_linear$pred$obs)
```

```
## [1] 0.19
```

```
svm_radial = train(
  y ~ .,
  data = train,
  method = "svmRadial",
  trControl = cv_ctrl,
  tuneLength = 10
)

svm_radial_roc = roc(predictor = as.integer(svm_radial$pred$pred),
                      response = svm_radial$pred$obs,
                      levels = c(0,1))

plot(svm_radial_roc)
```



```
auc(svm_radial_roc)
```

```
## Area under the curve: 0.9706
```

```
cat('Error Rate is ')
```

```
## Error Rate is
```

```
mean(svm_radial$pred$pred != svm_radial$pred$obs)
```

```
## [1] 0.03
```

For the SVM with radial kernel, the ROC curve is shown below and the AUC is 0.9698. The classification error rate is 0.03. Therefore, SVm with radial kernel performs better on the training data.

```
y1 = predict(svm_linear, test[,1:2])
y2 = predict(svm_radial, test[,1:2])
cat('The error rate of SVC is'); mean(as.integer(y1) != as.integer(test$y))
```

```
## The error rate of SVC is
```

```
## [1] 0.23
```

```
cat('the error rate of SVM with radial kernel is'); mean(as.integer(y2) != as.integer(test$y))
```

```
## the error rate of SVM with radial kernel is
```

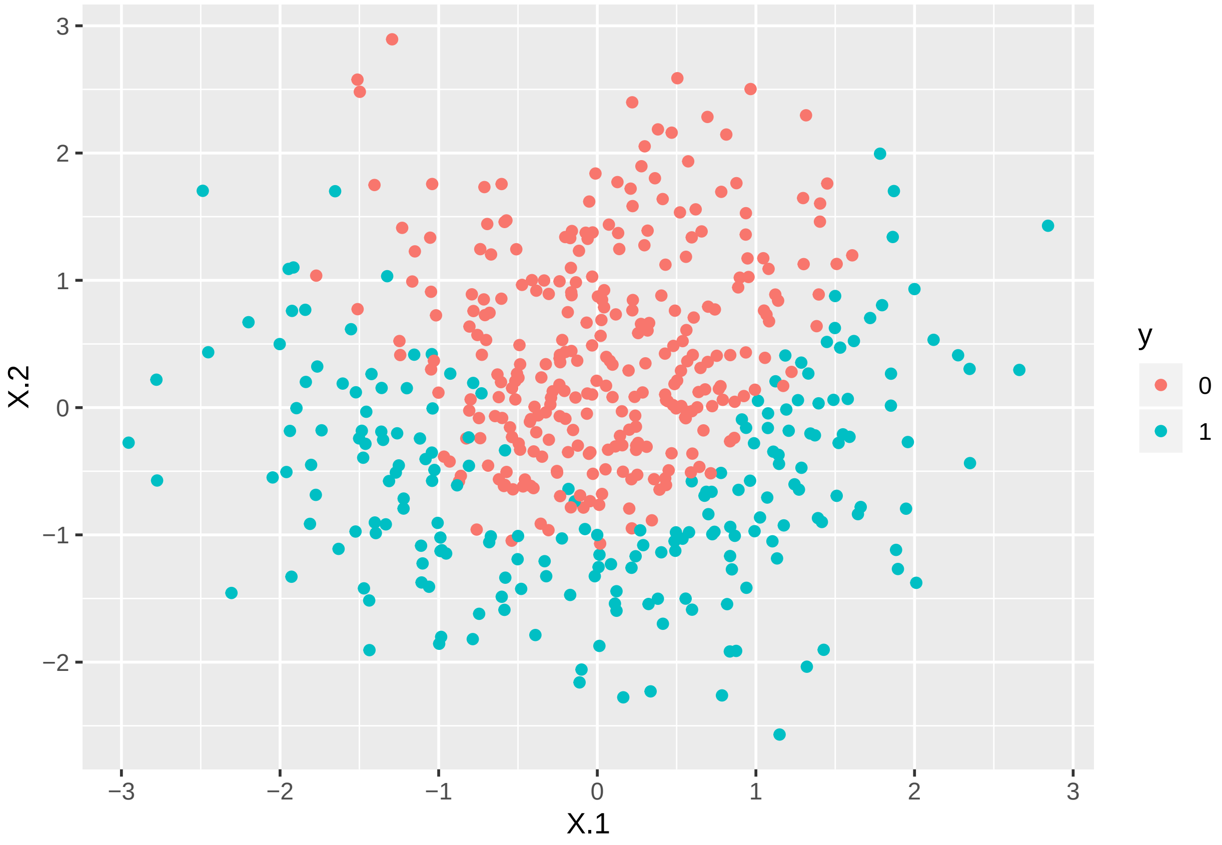
```
## [1] 0.05
```

Thus, SVM with raidal kernel performs better on the testing data.

SVM vs. logistic regression

Q2 & Q3

```
x1 = rnorm(500)
x2 = rnorm(500)
y = (x2 <= x1^2 - 1)
train = tibble(X.1 = x1, X.2 = x2, y = factor(as.numeric(y)))
z = runif(500, -0.3, 0.3)
train = train %>% mutate(
  X.1 = X.1 + z,
  X.2 = X.2 + z
)
train %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y))
```



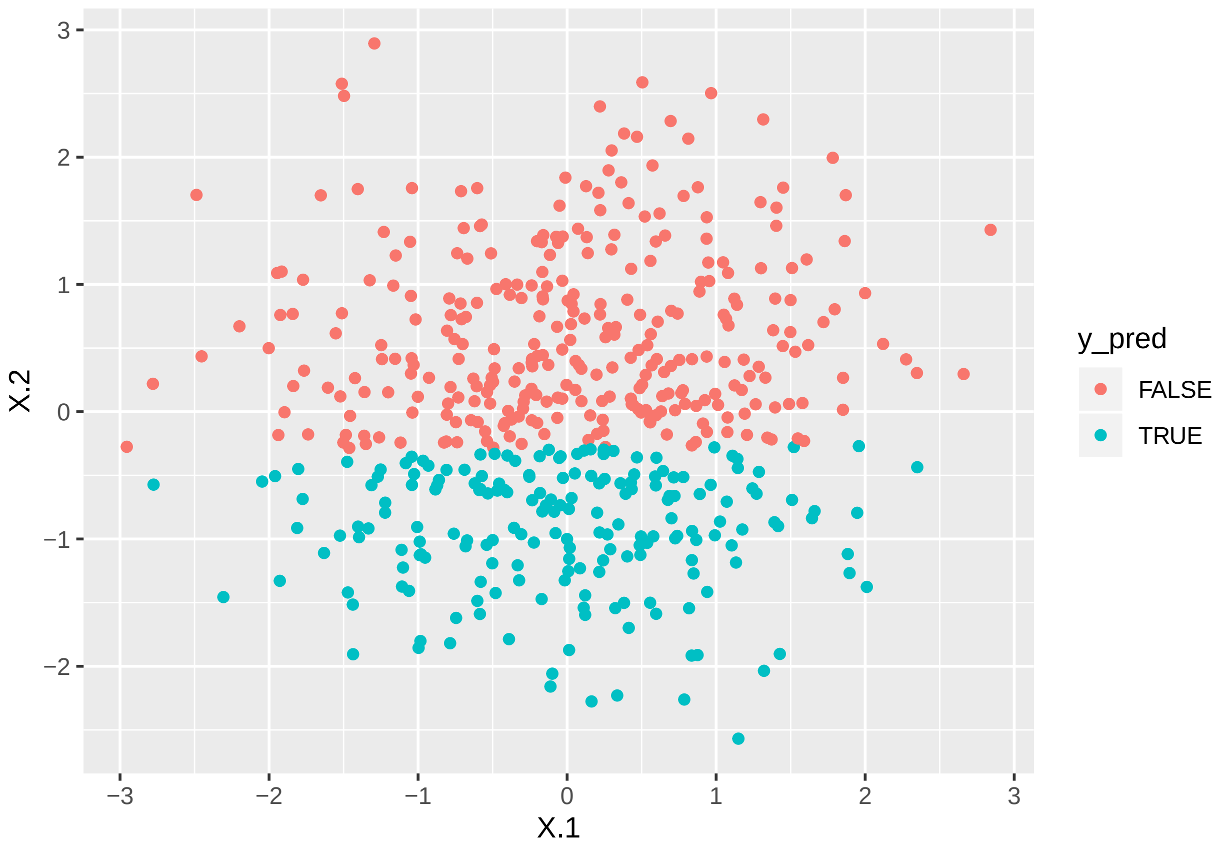
Q4

```
lgt_model = glm(y ~.,family=binomial(link='logit'),data=train)
```

Q5

```
y_pred = lgt_model$fitted.values > 0.5 %>% as.numeric()
train$y_pred = y_pred
```

```r
train %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y_pred))
```



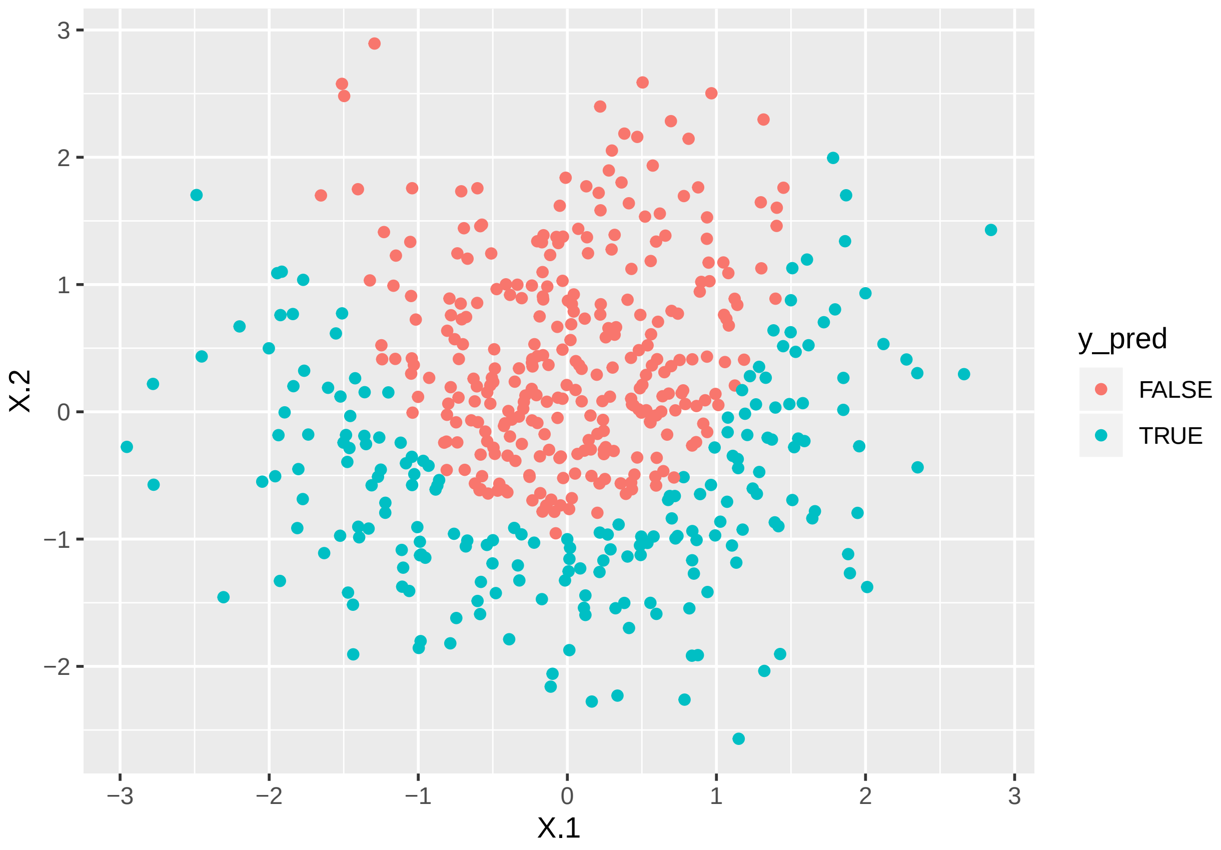
```r
train = train[,-4]
```

Q6

```r
train = train %>% mutate(
  X.12 = X.1 ^ 2,
  X.13 = X.1 ^ 3,
  X.22 = X.2 ^ 2,
  X.23 = X.2 ^ 3
)
lgt_model = glm(y ~ ., family=binomial(link='logit'), data=train)
```

Q7

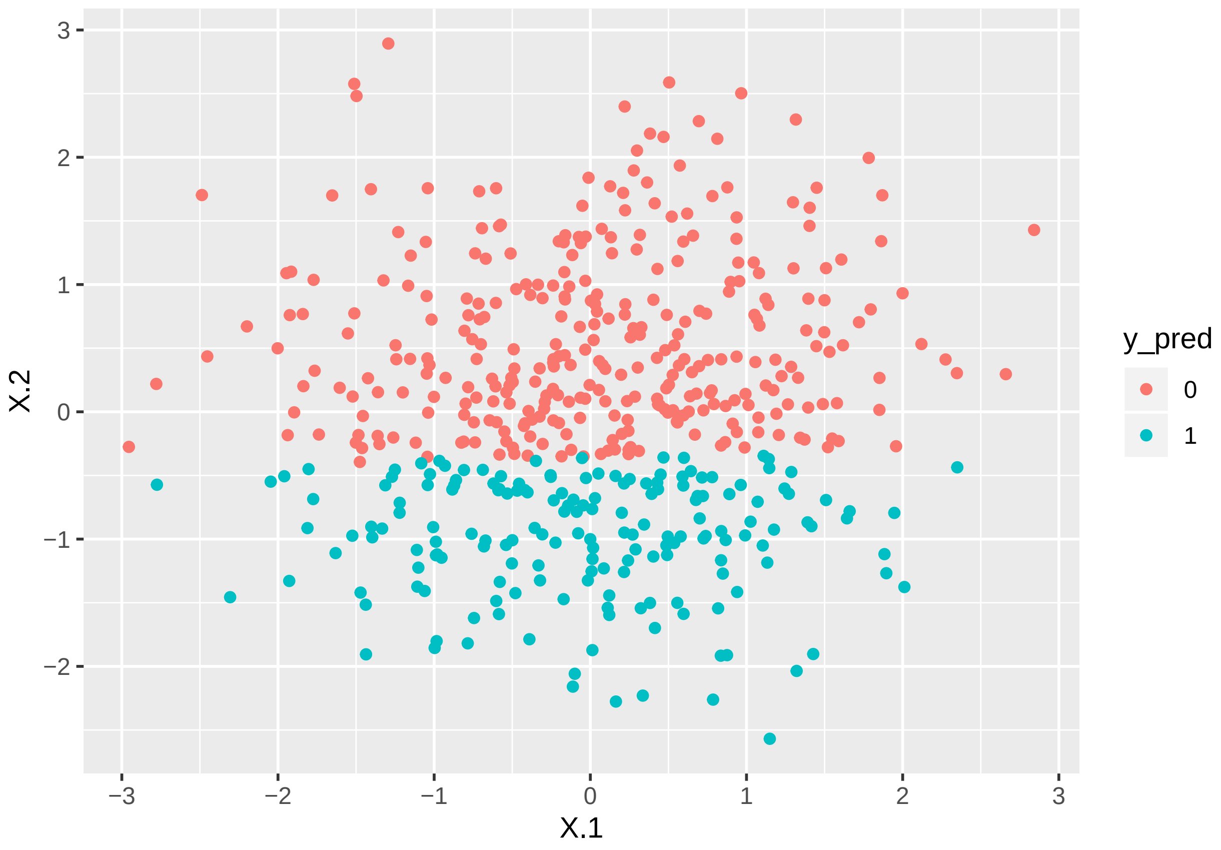
```r
y_pred = lgt_model$fitted.values > 0.5 %>% as.numeric()
train$y_pred = y_pred
train %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y_pred))
```



```r
train = train[,-c(4:8)]
```

Q8

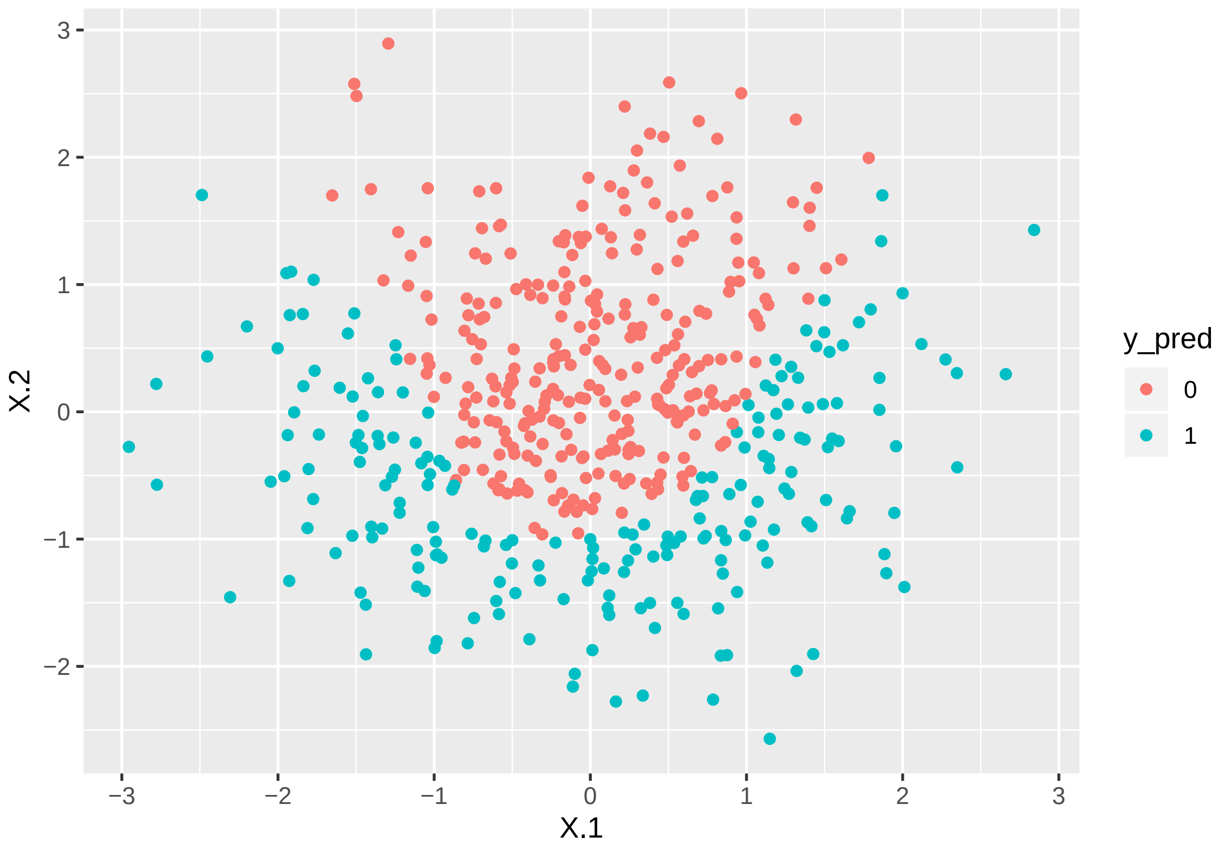
```r
svm_linear = train(
  y ~ .,
  data = train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10
)
y_pred = predict(svm_linear, train[,1:2])
train$y_pred = y_pred
train %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y_pred))
```



```
train = train[,-4]
```

Q9

```
svm_radial = train(
  y ~ .,
  data = train,
  method = "svmRadial",
  trControl = cv_ctrl,
  tuneLength = 10
)
y_pred = predict(svm_radial, train[,1:2])
train$y_pred = y_pred
train %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y_pred))
```



```r
train = train[,-4]
```

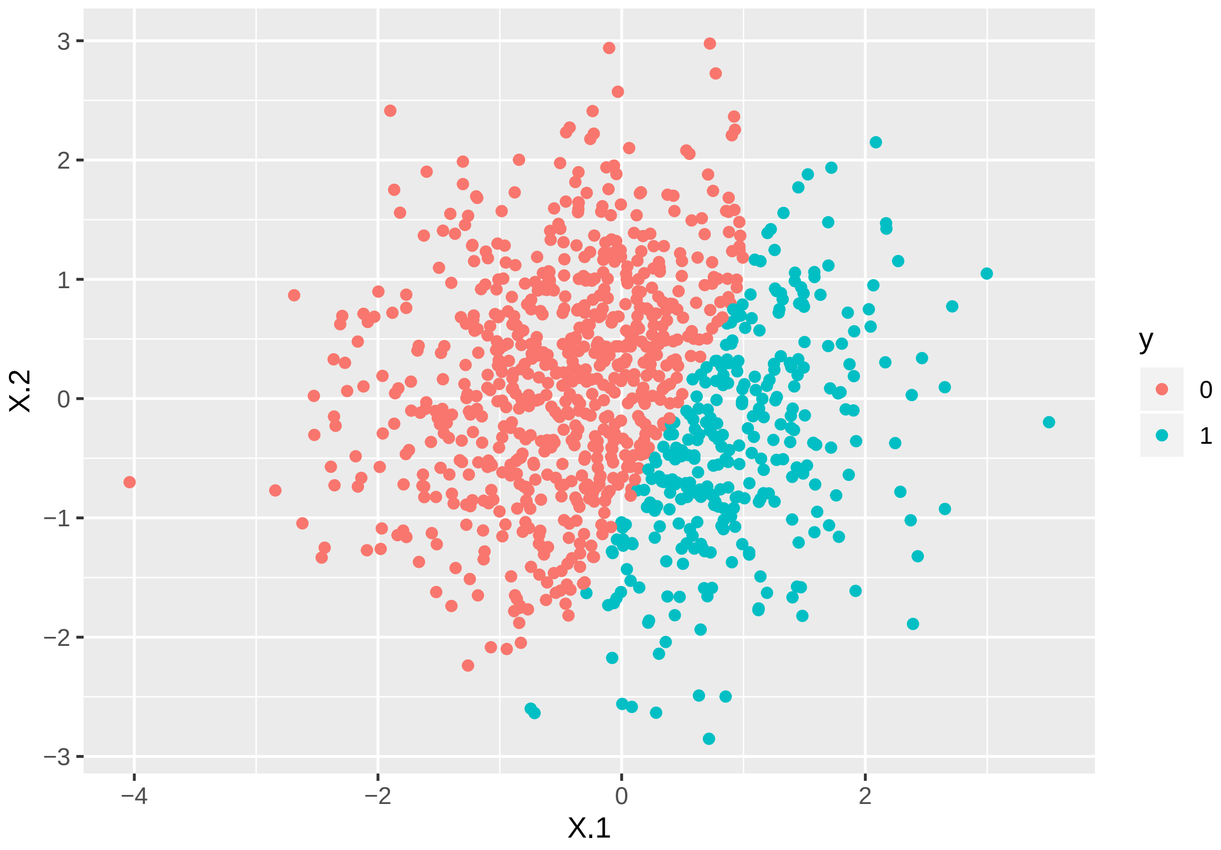
Q10

From the plots of each model, I find that both basic logistic regression and SVC show a linear decision boundary, which does not match the original data setting. However, the logistic regression with transformed features and the SVM with radial kernel have a nonlinear decision boundary, which is reasonable for the prediction job. The difference between those different approaches, is a trade-off between model complexity and interpretability. Linear relationship is more interpretable since it is monotonic and we can directly draw a conclusion between predictors. However, non-linear relationship, which is more complicated, sometimes does not offer a direct answer to how one predictor is related to another but can better approximate the true model in many cases. Thus, both SVM with radial kernel and tailored logistic regression give us a good approximation to the real model but they also bring about some trouble in model complexity.

Tuning Cost

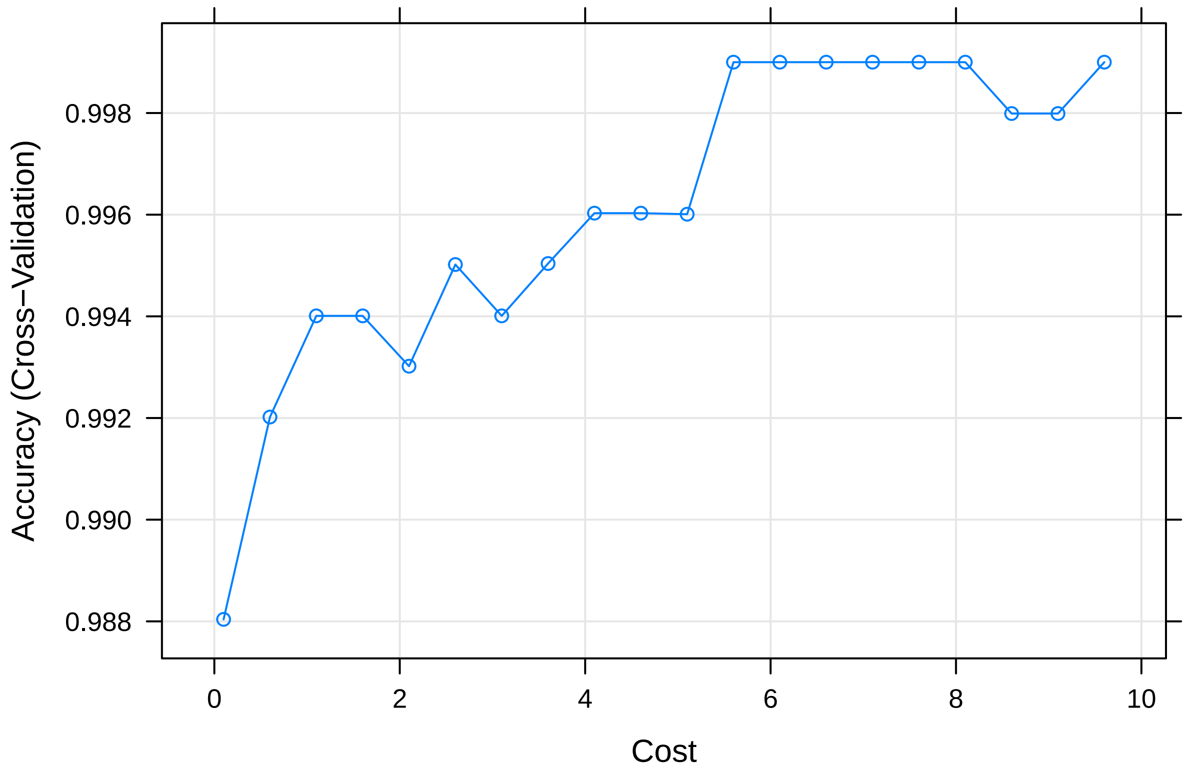
Q11

```r
set.seed(1233)
x1 = rnorm(1000)
x2 = rnorm(1000)
y = (x2 <= 2 * x1 - 1)
train = tibble(X.1 = x1, X.2 = x2, y = factor(as.numeric(y)))
train %>% ggplot(., aes(X.1, X.2)) +
  geom_point(aes(color = y))
```



Q12

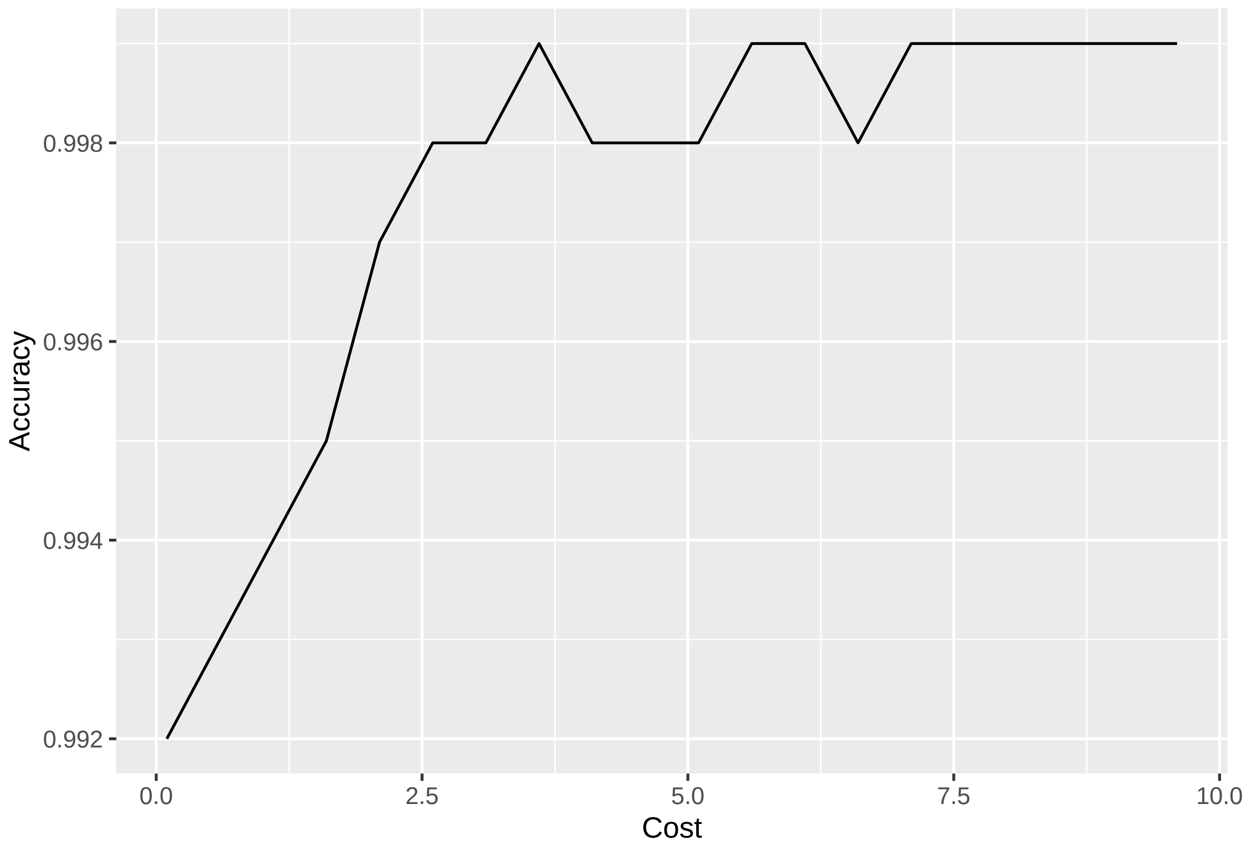
```
svm_linear = train(
  y ~ .,
  data = train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10,
  tuneGrid = expand.grid(C=seq(0.1, 10, 0.5))
)
plot(svm_linear)
```



As the Cost becomes larger, the accuracy of cross validation increases with fluctuation. When $C = 6$ we have the highest accuracy.

Q13

```
set.seed(123)
x1 = rnorm(1000)
x2 = rnorm(1000)
y = (x2 <= 2 * x1 - 1)
test = tibble(X.1 = x1, X.2 = x2, y = factor(as.numeric(y)))
C = seq(0.1, 10, 0.5)
err = c()
for (c in C) {
  svm_linear = train(
  y ~ .,
  data = train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10,
  tuneGrid = data.frame(.C = c))
  y_pred = predict(svm_linear, test[,1:2])
  err = c(err, 1-mean(y_pred != test$y))
}
df = tibble(Accuracy = err, Cost = C)
ggplot(df, aes(x = Cost, y = Accuracy)) +
  geom_line()
```



In the test case, I find that we have the highest accuracy when $C = 3.5$, which is different from the training case.

Q14

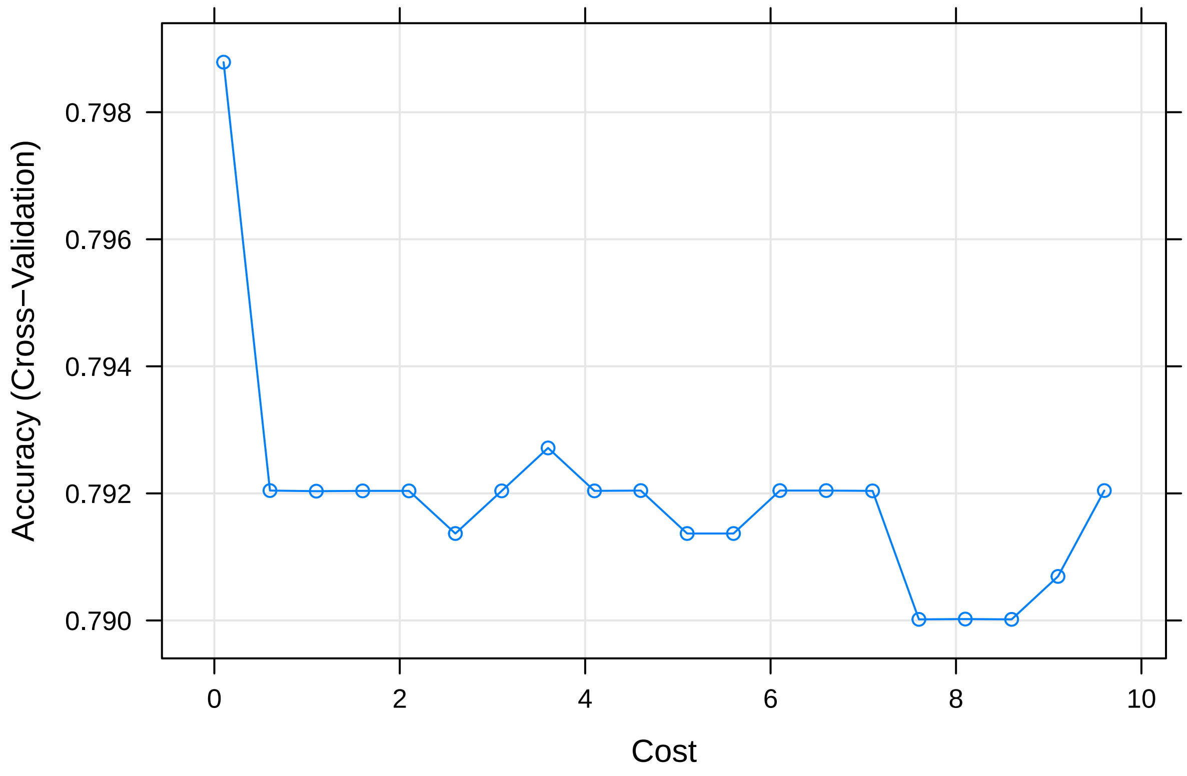
Both the trainig case and the testing case show that the accuracy of the model will incrase as C becomes larger. However, the training case needs a larger C to reach the best performance than the test case.

Application: Predicting attitudes towards racist college professors

Q15

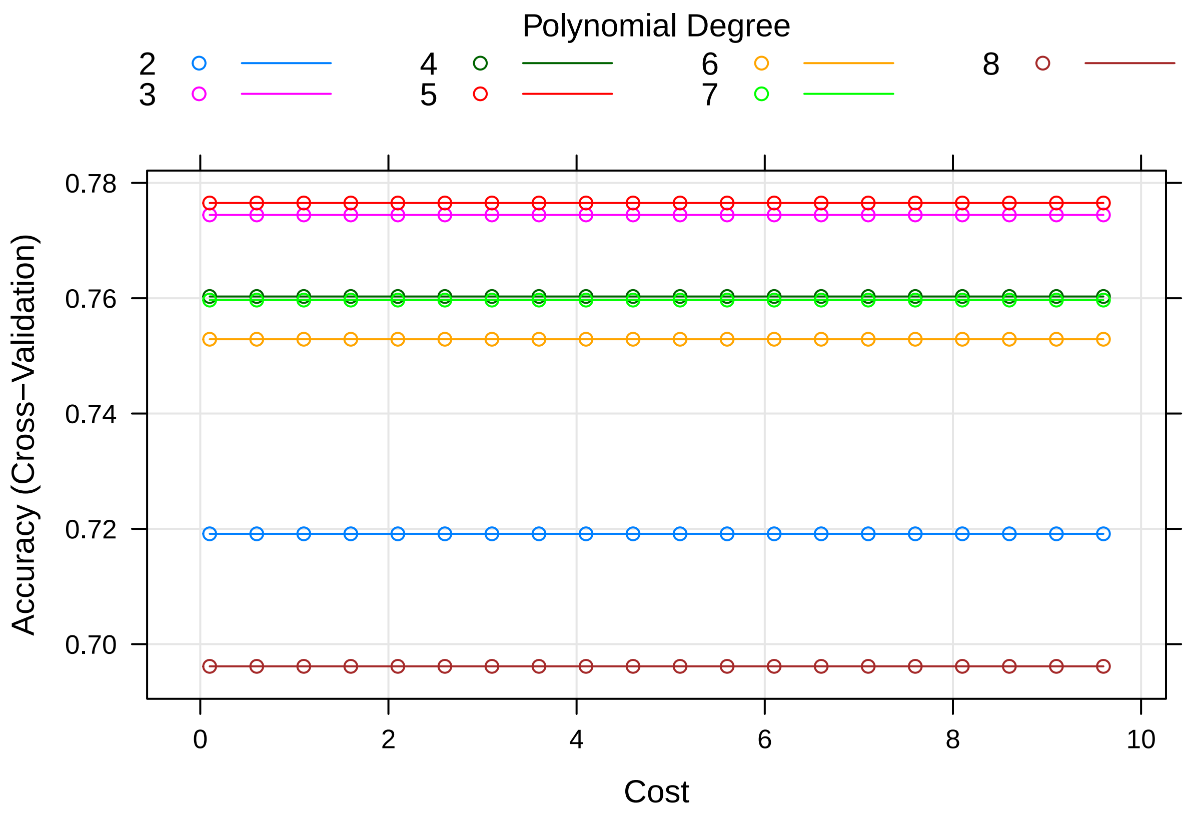
```r
train = read_csv('data/gss_train.csv')
test = read_csv('data/gss_test.csv')
train$colrac = as.factor(train$colrac)
test$colrac = as.factor(test$colrac)

svm_linear = train(
  colrac ~ .,
  data = train,
  method = "svmLinear",
  trControl = cv_ctrl,
  tuneLength = 10,
  tuneGrid = expand.grid(C=seq(0.1, 10, 0.5)))
plot(svm_linear)
```



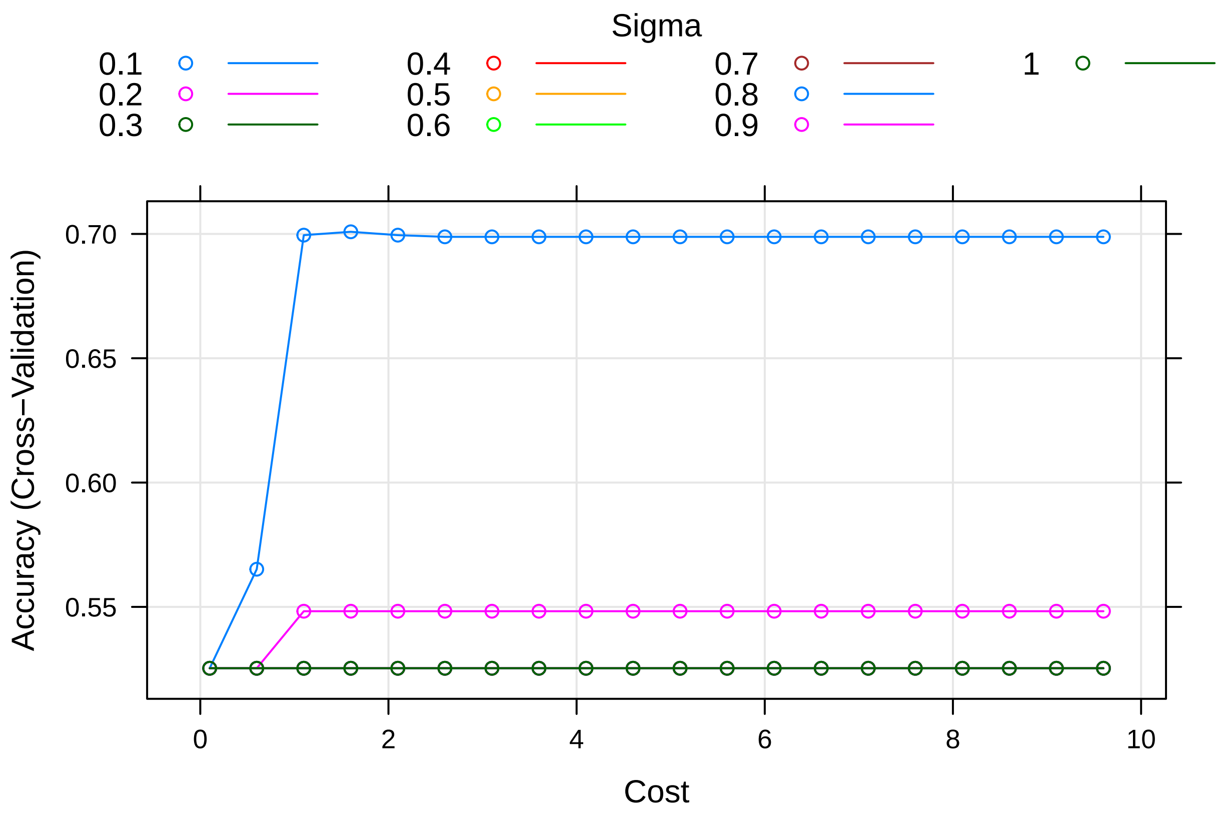
We have the largest accuracy when $C = 0.1$. From the figure, I find that either small Cost value or large Cost value can lead to low accuracy, which is different from my previous result in question 12 and question 13. ### Q16

```
svm_poly = train(
  colrac ~ .,
  data = train,
  method = "svmPoly",
  trControl = cv_ctrl,
  tuneGrid = expand.grid(C=seq(0.1, 10, 0.5),
                         degree = seq(2,8),
                         scale = 1)
)
plot(svm_poly)
```



For the polynomial SVM, C does not have a significant influence on the model accuracy but the degree does have. As the plot shows, we have the best model when $degree = 5$.

```r
svm_radial = train(
  colrac ~ .,
  data = train,
  method = "svmRadial",
  trControl = cv_ctrl,
  tuneGrid = expand.grid(C=seq(0.1, 10, 0.5),
                         sigma = seq(0.1,1, 0.1))
)
plot(svm_radial)
```



For the radial SVM, when $C = 1$ and $gamma = 0.8$ we have the highest accuracy. Among the three models, SVC model has the best performance ($Accuracy = 0.7935$) and the performance of polynomial SVM ($Accuracy = 0.785$) is close to SVC. However, the radial SVM does not perform well ($Accuracy = 0.7$), which might be due to that the range of parameters is not large enough.