Chen_Zhaoyang HW6

Zhaoyang Chen

3/8/2020

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(e1071)
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

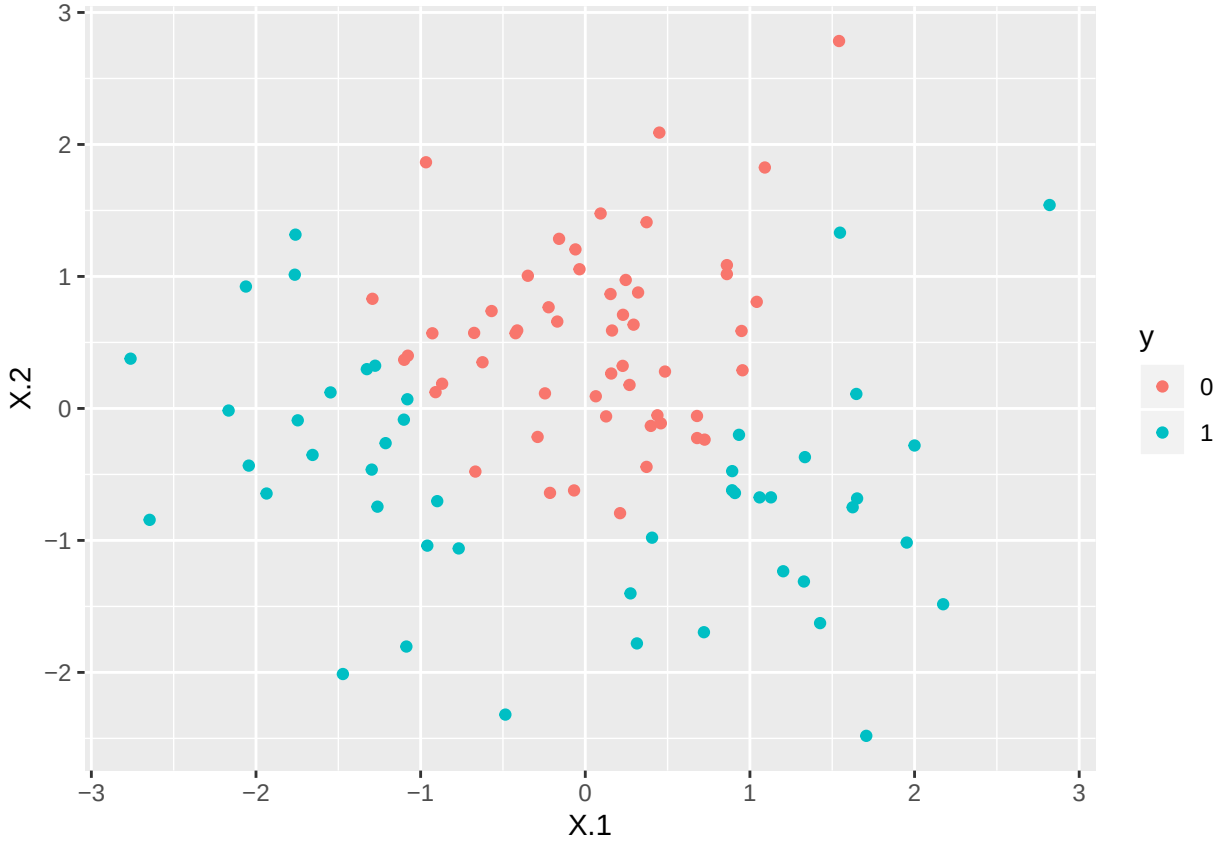
```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

Conceptual Exercises

Non-linear separation

```
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))
```




```
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.

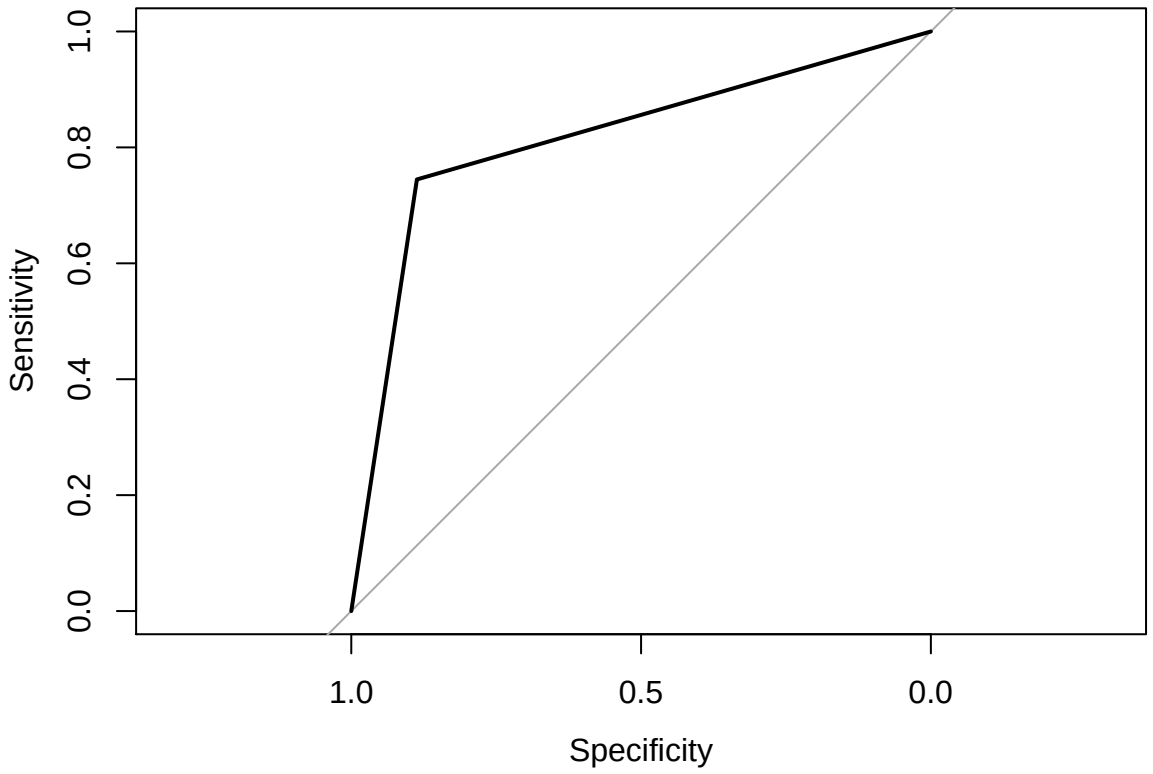
```
cv_ctrl = trainControl(method = "cv",
                       number = 10,
                       savePredictions = "final",
                       classProbs =F)

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))
```

```
## Setting direction: controls < cases
```

```
plot(svm_linear_roc)
```




```
auc(svm_linear_roc)
```

```
## Area under the curve: 0.8157
```

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

```
## [1] 0.18
```

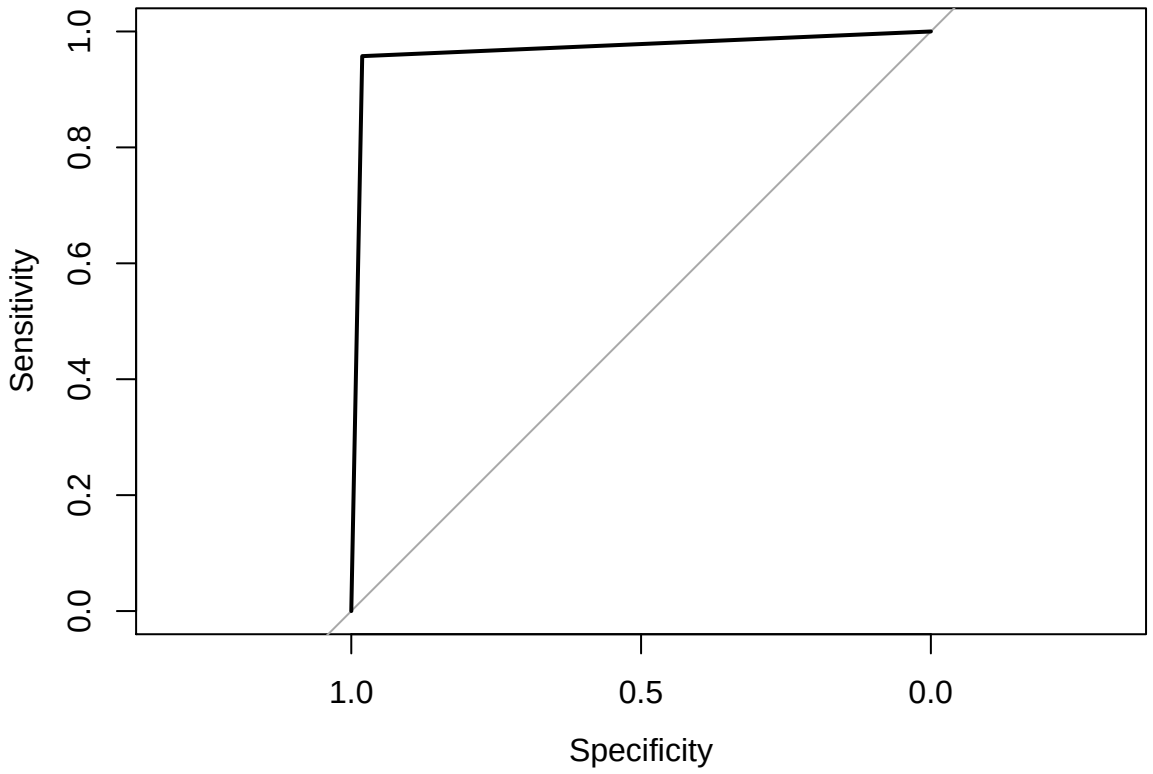
For the SVC, the ROC curve is shown below and the AUC is 0.8285. The classification error rate is 0.17.

```
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))
```

```
## Setting direction: controls < cases
```

```
plot(svm_radial_roc)
```




```
auc(svm_radial_roc)
```

```
## Area under the curve: 0.9693
```

```
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])
mean(as.integer(y1) != as.integer(test$y))
```

```
## [1] 0.28
```

```
mean(as.integer(y2) != as.integer(test$y))
```

```
## [1] 0.02
```

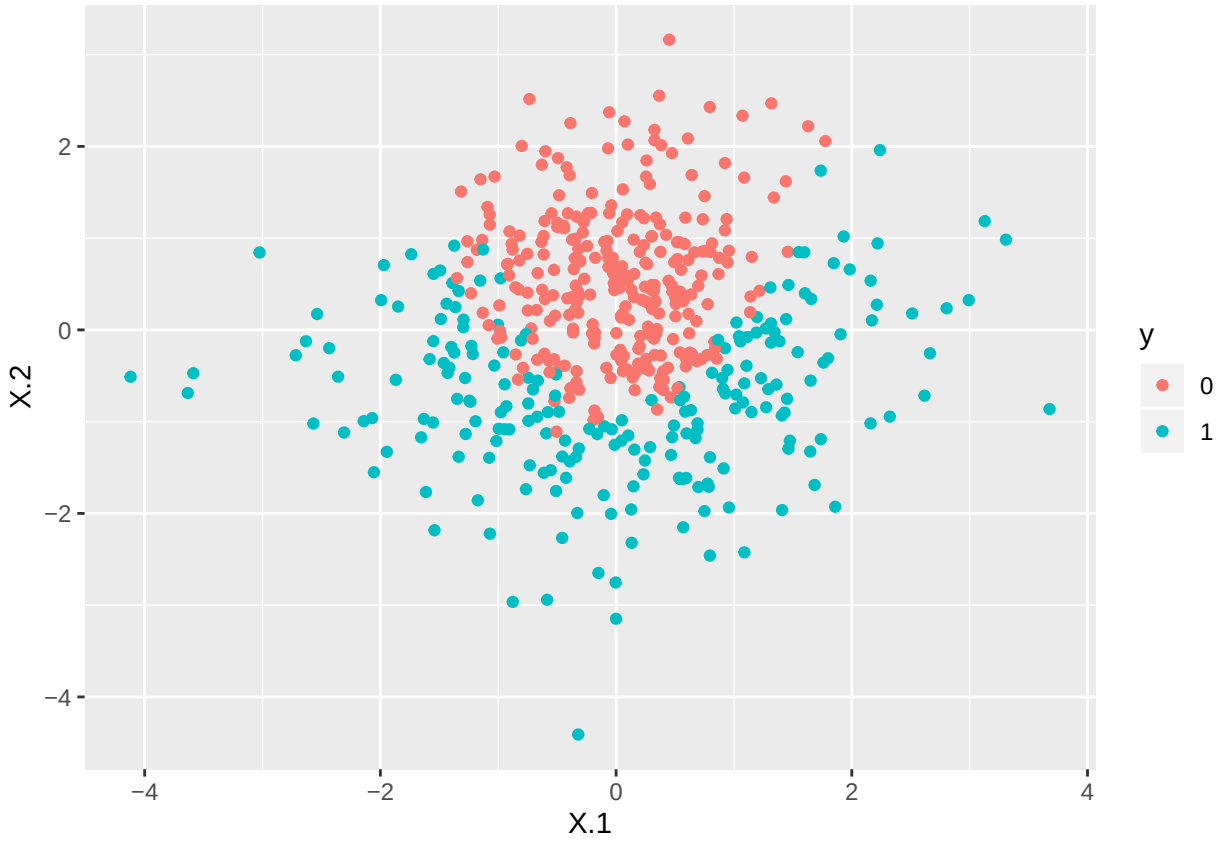
The error rate of SVC is 0.3 and the error rate of SVM with radial kernel is 0.07. Thus, SVM with raidal kernel performs better on the testing data.

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))
```

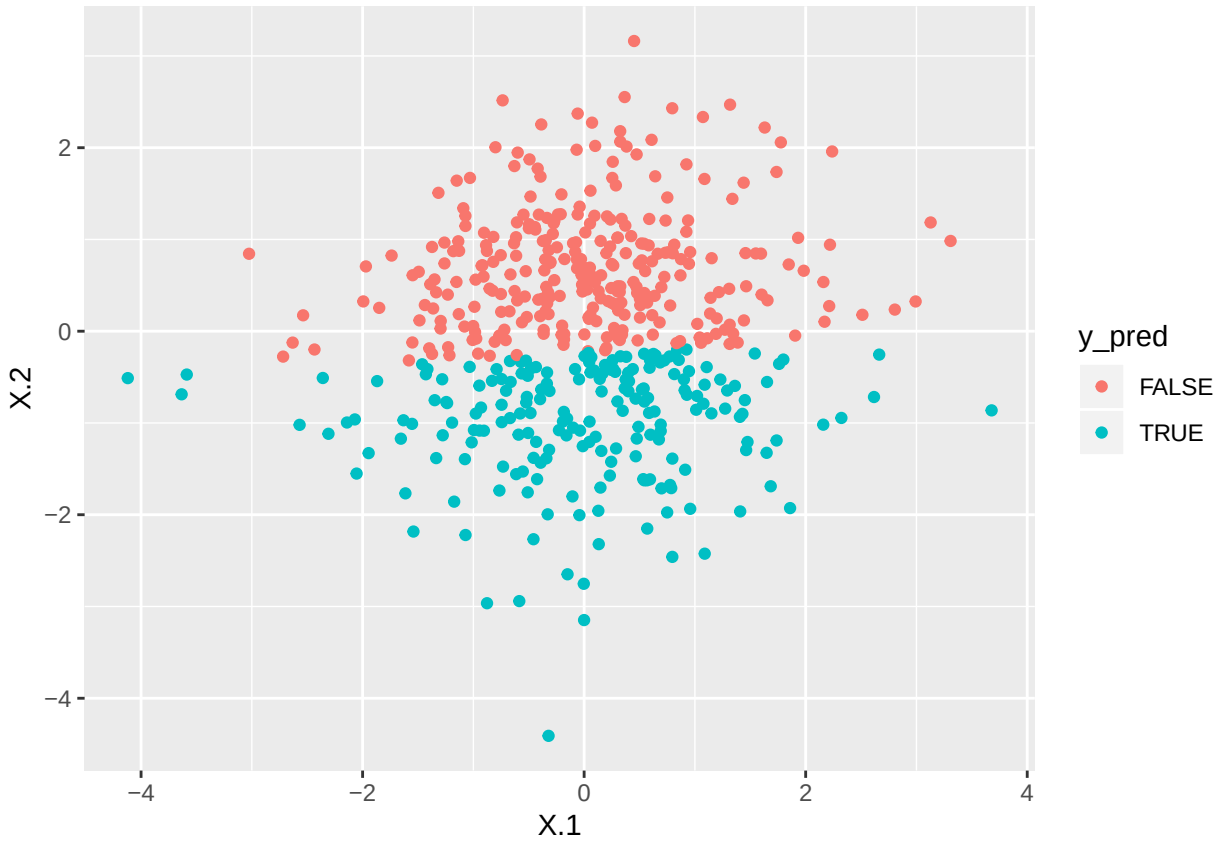


Q4

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




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

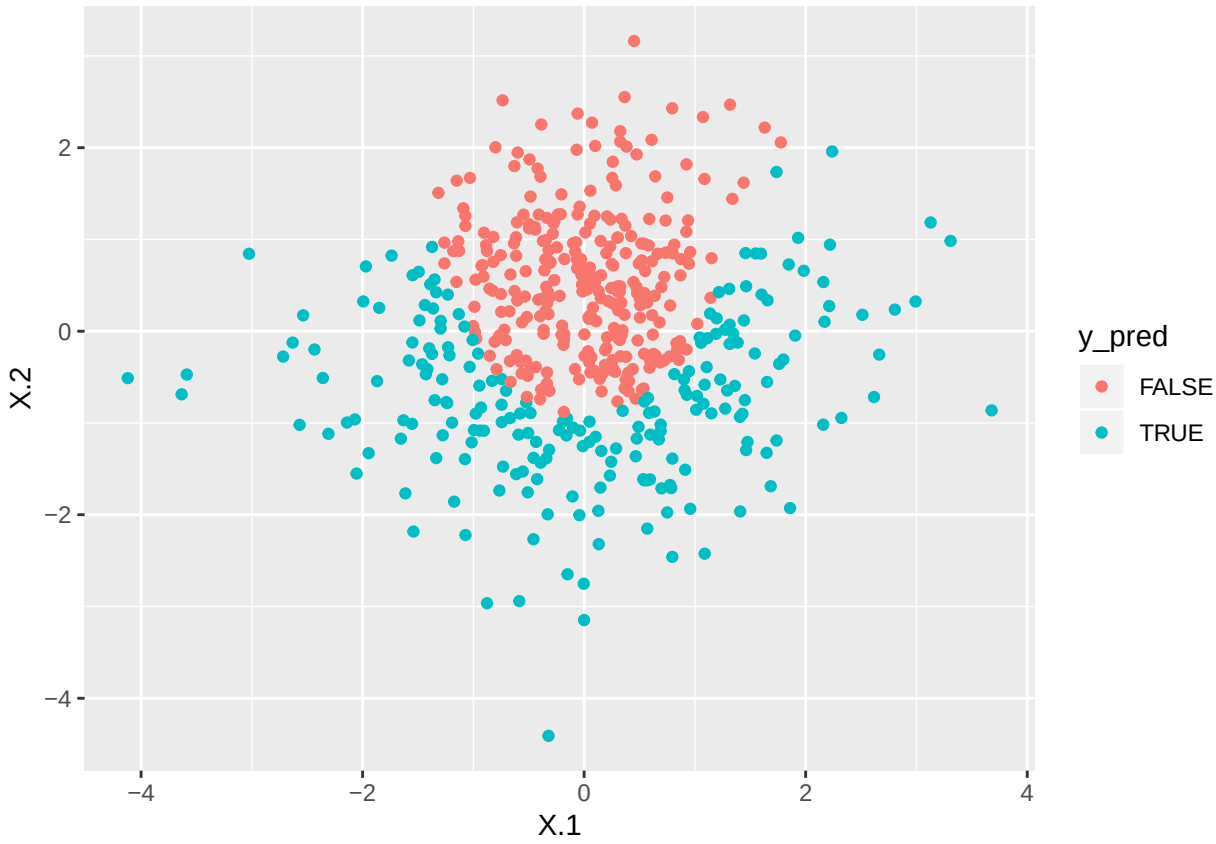
Q6

```
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)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Q7

```
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))
```




```
train = train[,-c(4:7)]
```

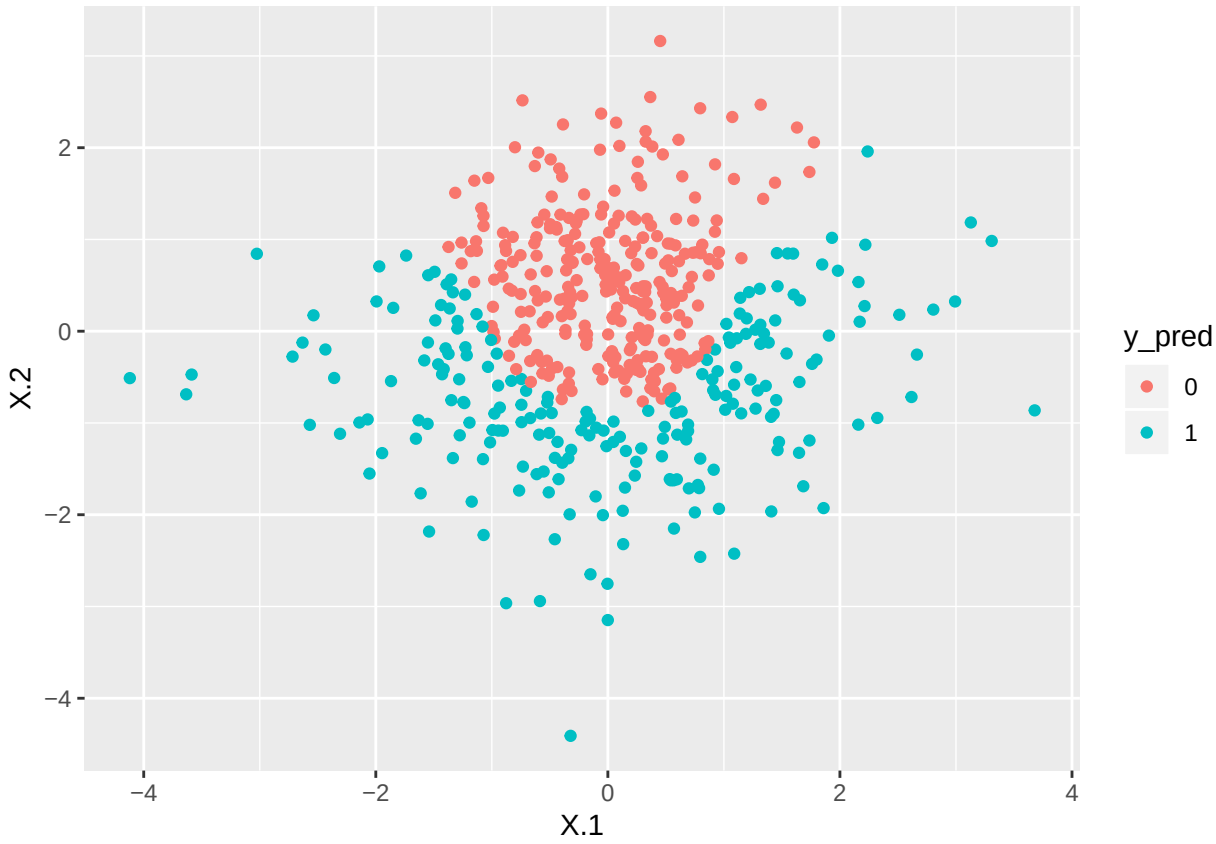
Q8

```
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))
```




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