Homework 5

Phoebe Parrish

3/13/2020

Question 1

Suppose we produce 10 bootstrapped samples from a data set containing red and green classes. We then fit a classification tree to each bootstrapped sample and, for a specific value of X, produce ten estimates of $P(\text{Class is Red} \mid X)$:

$$0.01, 0.01, 0.05, 0.1, 0.51, 0.6, 0.6, 0.65, 0.66, 0.67$$

There are two commonly-used approaches to combine these results together into a single class prediction. One is the majority vote approach: namely, we assign to class red if more than half of the estimates of $P(\text{Class is Red} \mid X)$ exceed 0.5, and we assign to class green otherwise. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

```r
p.red <- c(0.01, 0.01, 0.05, 0.1, 0.51, 0.6, 0.6, 0.65, 0.66, 0.67)
p.red

# Approach 1
n.over.half <- sum(p.red > 0.5)
which.class.1 <- ifelse(n.over.half > (length(p.red)/2), "red", "green")

# Approach 2
mean.p.red <- mean(p.red)
which.class.2 <- ifelse(mean.p.red > 0.5, "red", "green")

# make output df and print results
q1.out.df <- tibble("Approach"=c(1,2),
                    "Predicted.Class"=c(which.class.1, which.class.2))
kable(q1.out.df)
```

```
##  [1] 0.01 0.01 0.05 0.10 0.51 0.60 0.60 0.65 0.66 0.67
```

Approach	Predicted.Class
1	red
2	green

Question 2

Find a data set of your choice that consists of a $n \times p$ matrix X and a qualitative response Y with K classes. If there are any qualitative variables, then you should either toss them, or else re-code using dummy variables. Make sure that you have at least 10 observations in each class, and at least 3 features.

(A) What are the values of n, p, K? Describe the data. (Where did you find it? what do the features represent? what do the response classes mean? etc.)

(B) Make a plot displaying the n observations projected onto the first 2 principal components. Color the observations according to their class label (make sure to include a legend, to label axes, etc.). What proportion of variance is explained by the first two principal components?

(C) Now cluster the data matrix X using K-means clustering (where here you set K to be the true number of classes). Repeat the plot from (B), but this time color the observations according to the cluster labels.

(D) Use the `table()` function in `R` to make a contingency table displaying the true class labels versus the cluster labels. Use the adjusted Rand Index to quantify the extent to which the true class labels agree with the cluster labels. Comment on your results.

(A) For this problem, I will be using the Cervical Cancer (Risk Factors) dataset from the UCI Machine Learning Repository. The dataset was collected at Hospital Universitario de Caracas in Venezuela. Some patients chose not to answer questions due to privacy concerns (NA values). Dataset descriptors include:

- $n = 858$ individuals
- $p = 8$ features that describe each patient, their risk factors, and their cancer diagnosis
- $K = 2$ classes

Features

1. **age:** Quantitative variable representing each patient's age.
2. **num.sexual.partners:** Quantitative variable representing each patient's number of sexual partners.
3. **num.pregnancies:** Quantitative variable representing the number of pregnancies for each patient.
4. **smokes.yrs:** Quantitative variable representing the number of years that the patient has smoked cigarettes.
5. **hormonal.contraceptives.yrs:** Quantitative variable representing the number of years that the patient has taken hormonal contraceptives.
6. **IUD.yrs:** Quantitative variable representing the number of years the patient has used an IUD (intrauterine device) as a contraceptive.
7. **num.STDs:** Quantitative variable representing the number of STDs the patient has been diagnosed with.
8. **cancer.Dx:** Binary variable representing whether the patient has been diagnosed with cervical cancer.

Response classes
The response variable in this dataset is **cancer.Dx**. The response classes are:

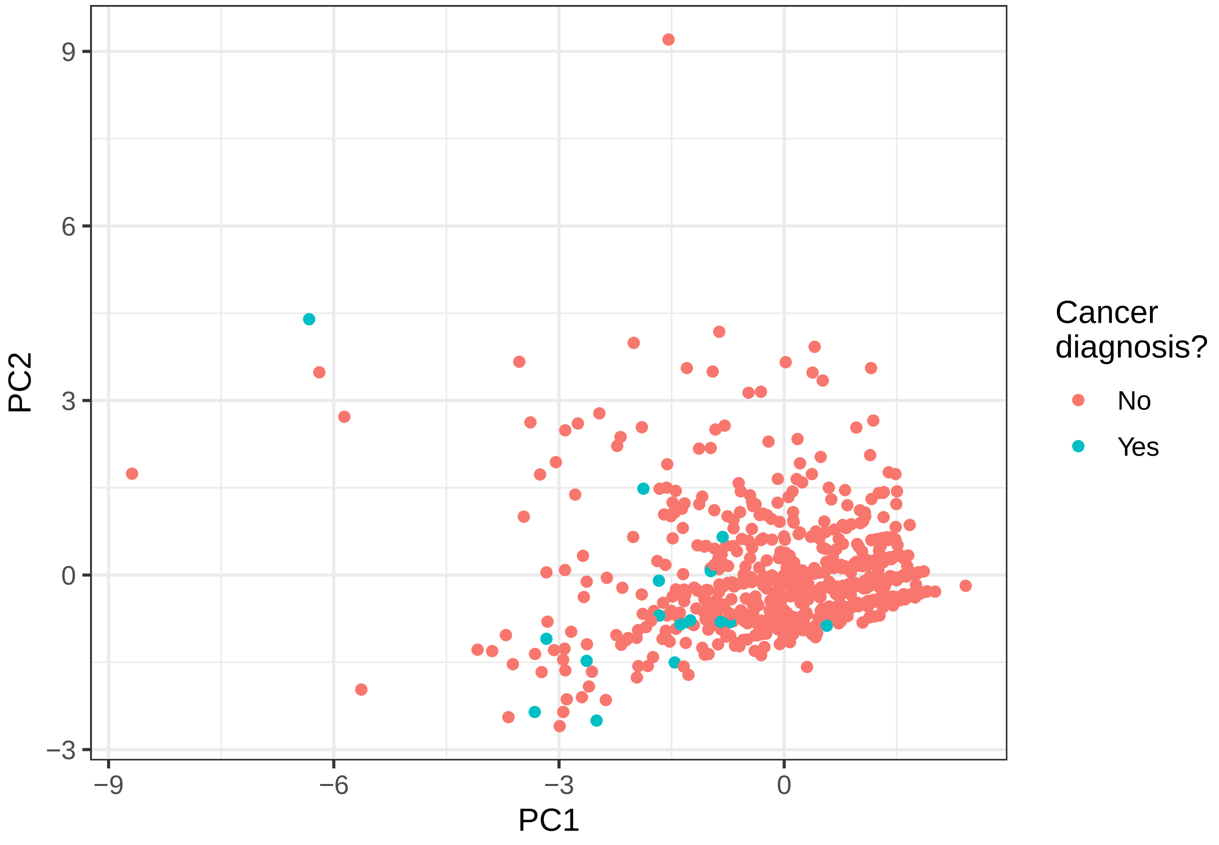
- **0:** No cervical cancer diagnosis.
- **1:** Yes, the patient has been diagnosed with cervical cancer.

```r
# read in the cervical cancer dataset, set "?" to NA
cancer.df <- read.csv("../input_data/risk_factors_cervical_cancer.csv", header = TRUE,
                      na.strings="?", strip.white=TRUE)

# keep only a subset of columns with quantitative variables,
#   rename variables in a nicer format
cancer.df <- cancer.df %>%
  dplyr::select(c("Age", "Number.of.sexual.partners", "Num.of.pregnancies",
                  "Smokes..years.", "Hormonal.Contraceptives..years.",
                  "IUD..years.", "STDs..number.", "Dx.Cancer")) %>%
  dplyr::rename(age="Age", num.sexual.partners="Number.of.sexual.partners",
                num.pregnancies="Num.of.pregnancies",
                smokes.yrs="Smokes..years.",
                hormonal.contraceptives.yrs="Hormonal.Contraceptives..years.",
                 IUD.yrs="IUD..years.", num.STDs="STDs..number.", cancer.Dx="Dx.Cancer")

# drop NA values
cancer.df <- na.omit(cancer.df)

# print number of observations in each response class
n.per.class.df <- cancer.df %>% group_by(cancer.Dx) %>% summarize(n=n())
kable(n.per.class.df)
```

cancer.Dx	n
0	656
1	17

(B)
Plotting the first two PCs:

```
# run PCA on the cancer.df
pr.out <- prcomp(cancer.df, scale=TRUE)

# plot first two PCs
pr.out.df <- as_tibble(pr.out$x) %>% mutate(cancer.Dx=as.factor(cancer.df$cancer.Dx))

ggplot(pr.out.df, aes(x=PC1, y=PC2, color=cancer.Dx)) + geom_point() +
  theme_bw(base_size=12) + scale_color_discrete(name="Cancer\ndiagnosis?",
                                                labels=c("No", "Yes"))
```



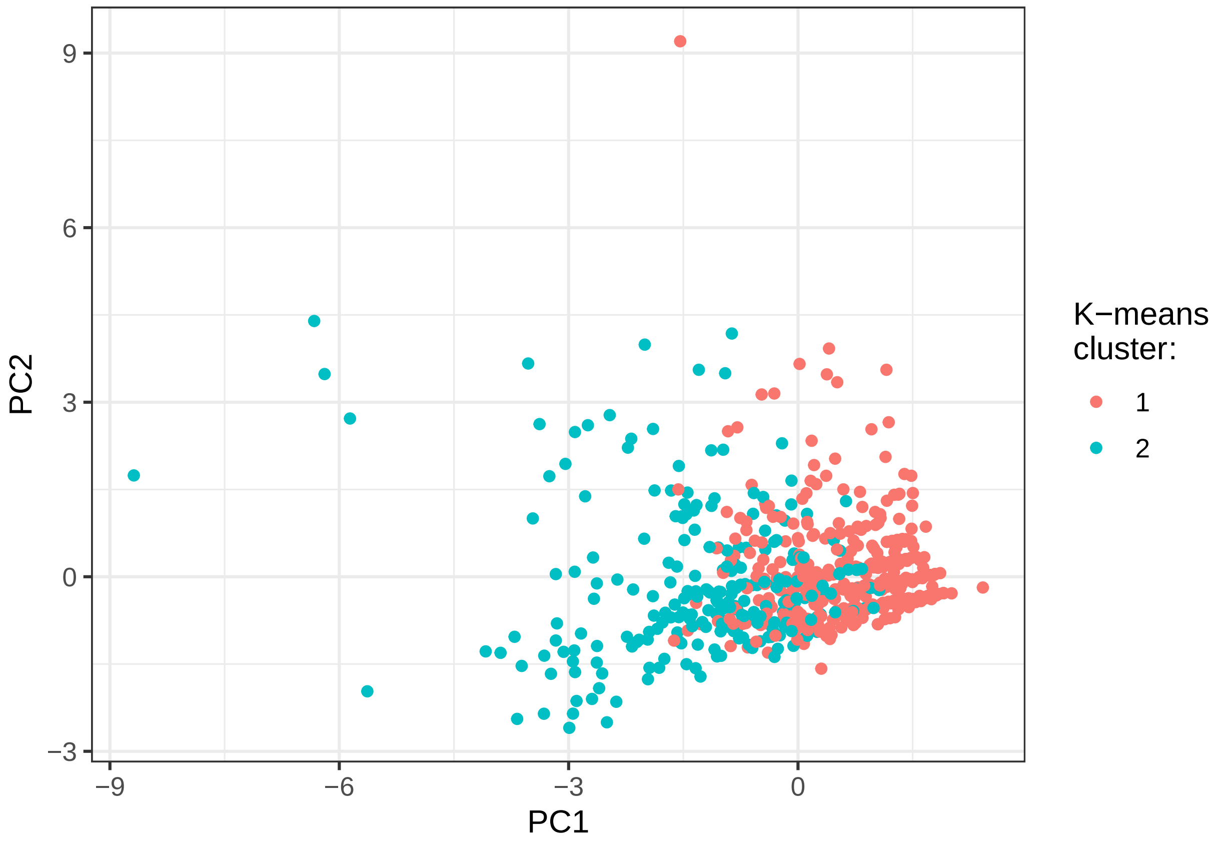
Calculating proportion of variance explained by the first two PCs:

```
# calculate proportion of variance explained
pr.var <- pr.out$sdev^2
pve <- pr.var/sum(pr.var)

pve.out.df <- tibble(PC=c(1,2), PVE=c(pve[1], pve[2]))
kable(pve.out.df)
```

PC	PVE
1	0.2429706
2	0.1394059

(C)

```
set.seed(123)

km.out <- kmeans(cancer.df, centers=2)
km.out.df <- pr.out.df %>% mutate(km.cluster=as.factor(km.out$cluster), cancer.Dx=cancer.df$cancer.Dx)
#km.out.df

ggplot(km.out.df, aes(x=PC1, y=PC2, color=km.cluster)) + geom_point() +
  theme_bw(base_size=12) + scale_color_discrete(name="K-means\ncluster:")
```



(D)

```
set.seed(123)

# make contingency table comparing cancer diagnosis to K-means cluster number
cont.table <- table(km.out.df$cancer.Dx, km.out.df$km.cluster)
cat("Contingency table:")
```

```
cont.table
```

```
## Contingency table:
##       1   2
##   0 411 245
##   1   6  11
```

```
# calculate Average Rand Index
ari.out <- ari(cont.table)
cat("\n")
```

```
ari.out
```

```
##     Adjusted Rand Index (alpha = 0.05)
##
## ARI                  = 0.01 (poor recovery)
## Confidence interval  = [0, 0.02]
##
## p-values:
##   * Qannari test     = < 0.001
##   * Permutation test =   0.001
```

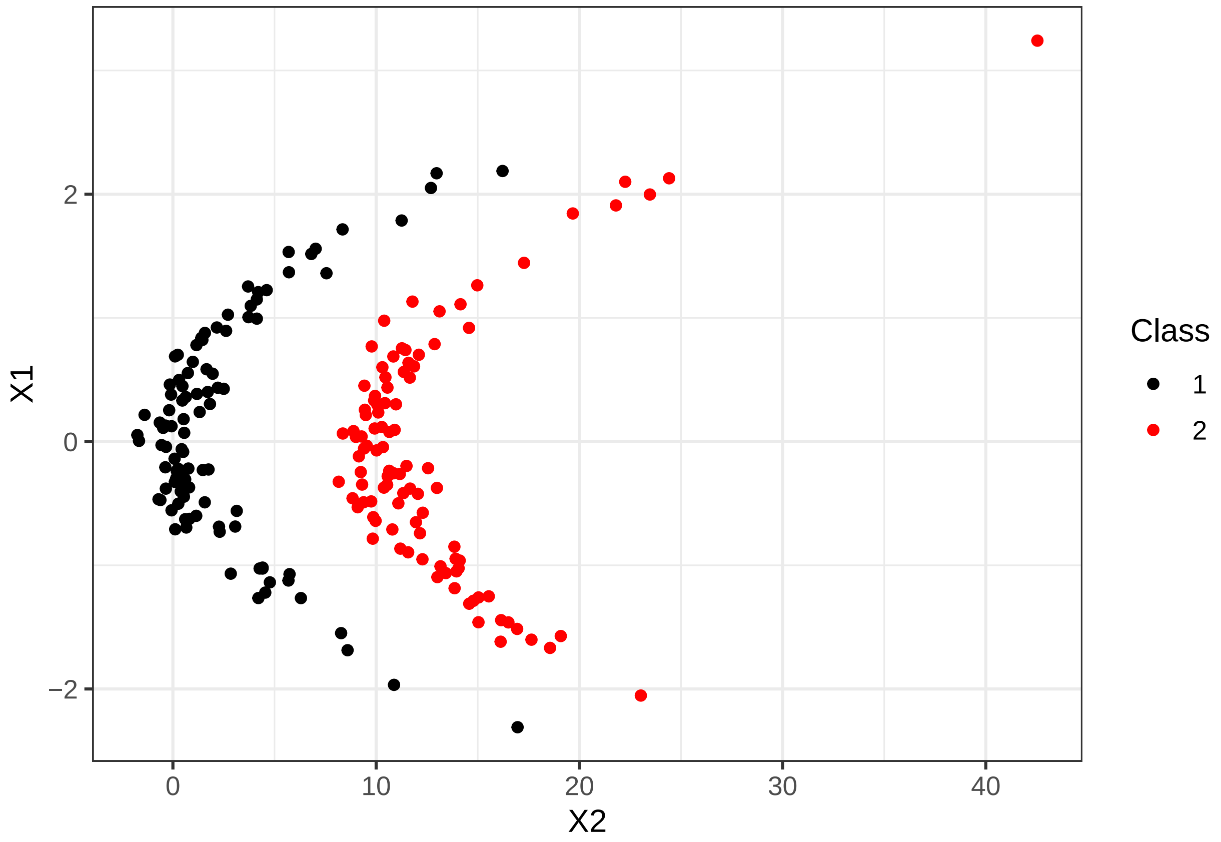
Here, the Adjusted Rand Index equals 0.01, which suggests very poor classification using K-means clustering. This result makes sense given that the two classes were not well-separated based on the first two principal components calculated from the dataset. This may be because the features collected are not really that predictive of the patient's cancer risk.

Question 3

Simulate a two-class data set with 100 training observations and 100 test observations, and two features, in which there is a visible but non-linear separation between the two classes. Show that in this setting, a support vector machine with a polynomial kernel (with degree greater than 1) or a radial kernel will outperform a support vector classifier on the training data. Which technique performs best on the test data? Make plots and report training and test error rates in order to back up your assertions.

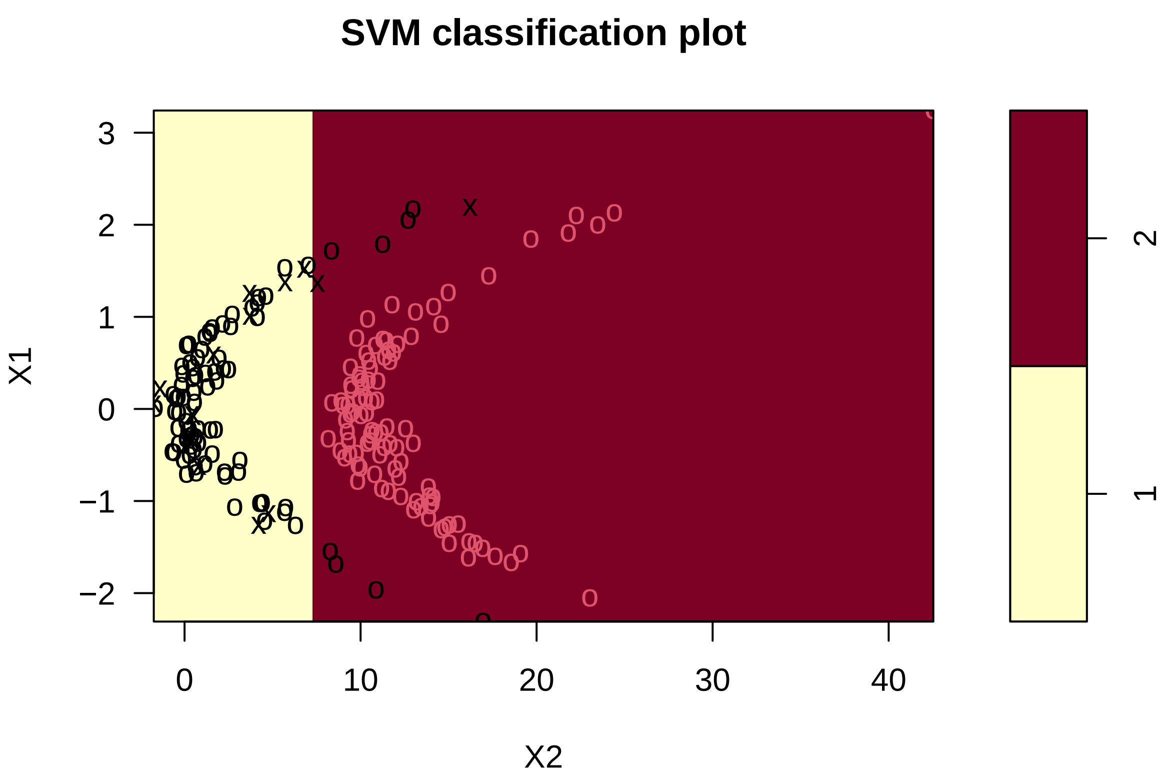
Generate and plot dataset:

```
set.seed(123)

# make two features, with X2 being a quadratic function of X1
sim.X1 <- rnorm(200)
sim.X2 <- 3 * sim.X1^2 + rnorm(200)

# add 10 to sim.X2 to separate the classes
sim.X2[101:200] <- sim.X2[101:200] + 10

# generate class labels (Y)
sim.Y <- c(rep(1, 100), rep(2, 100))

# make dataframe, separate into training and test sets, and plot
sim.df <- tibble(Y=as.factor(sim.Y), X1=sim.X1, X2=sim.X2)

sim.df.mod <- rowid_to_column(sim.df, "ID")
sim.train <- sim.df.mod %>% sample_frac(0.5)
sim.test <- anti_join(sim.df.mod, sim.train, by="ID")

sim.train <- sim.train %>% dplyr::select(-ID)
sim.test <- sim.test %>% dplyr::select(-ID)

ggplot(sim.df, aes(x=X2, y=X1, color=Y)) + geom_point() +
  theme_bw(base_family="Helvetica", base_size=12) +
  scale_color_manual(name="Class", values=c("black", "red"))
```



Run SVM with a linear kernel

```
svm.linear.fit <- svm(Y ~ ., data=sim.train, kernel="linear", cost=10, scale=FALSE)

# plot
plot(svm.linear.fit, sim.df)
```

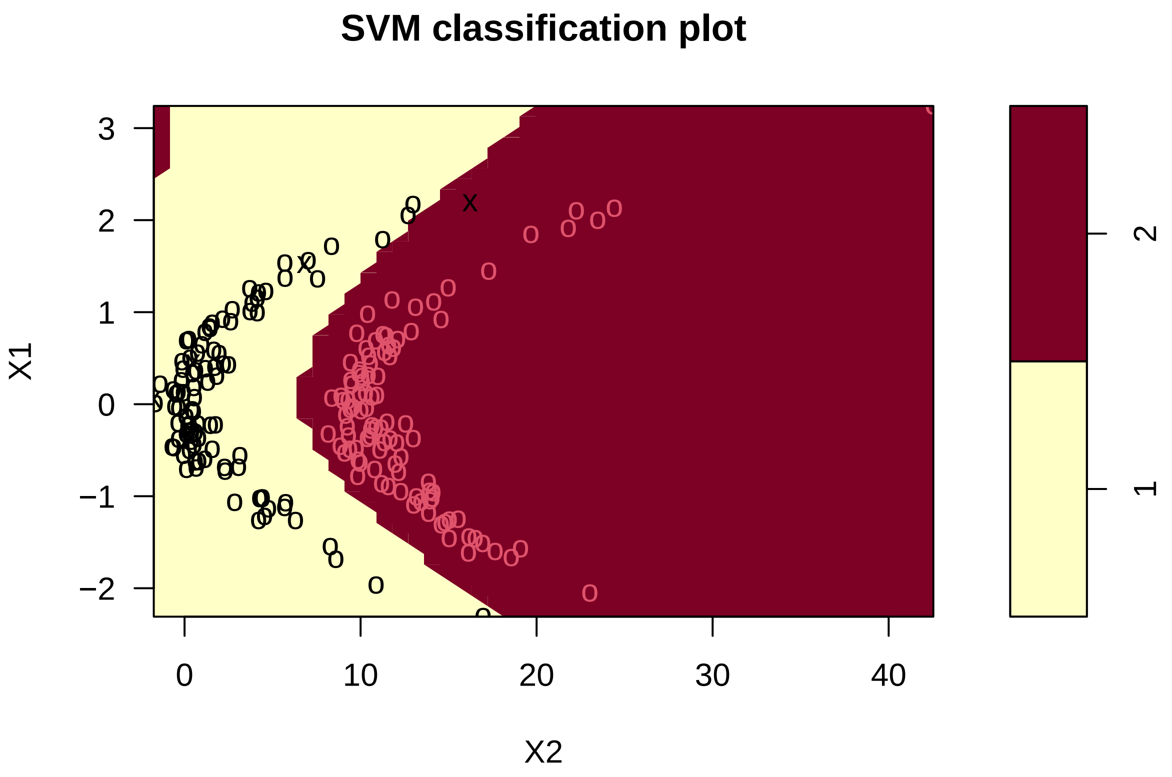


```r
# calculate training error
svm.linear.train.df <- tibble(Y=sim.train$Y, Y.hat=predict(svm.linear.fit, sim.train)) %>%
  mutate(error=ifelse(Y==Y.hat, 0, 1))
svm.linear.train.err <- sum(svm.linear.train.df$error)/nrow(svm.linear.train.df)
cat("Training error rate =", svm.linear.train.err)

# calculate test error
svm.linear.test.df <- tibble(Y=sim.test$Y, Y.hat=predict(svm.linear.fit, sim.test)) %>%
  mutate(error=ifelse(Y==Y.hat, 0, 1))
svm.linear.test.err <- sum(svm.linear.test.df$error)/nrow(svm.linear.test.df)
cat("\nTest error rate =", svm.linear.test.err)
```

```
## Training error rate = 0.03
## Test error rate = 0.07
```

Run SVM with a polynomial kernel

```r
svm.poly.fit <- svm(Y ~ ., data=sim.train, kernel="polynomial", cost=10, scale=FALSE)

# plot
plot(svm.poly.fit, sim.df)
```



```r
# calculate training error
svm.poly.train.df <- tibble(Y=sim.train$Y, Y.hat=predict(svm.poly.fit, sim.train)) %>%
  mutate(error=ifelse(Y==Y.hat, 0, 1))
svm.poly.train.err <- sum(svm.poly.train.df$error)/nrow(svm.poly.train.df)
cat("Training error rate =", svm.poly.train.err)

# calculate test error
svm.poly.test.df <- tibble(Y=sim.test$Y, Y.hat=predict(svm.poly.fit, sim.test)) %>%
  mutate(error=ifelse(Y==Y.hat, 0, 1))
svm.poly.test.err <- sum(svm.poly.test.df$error)/nrow(svm.poly.test.df)
cat("\nTest error rate =", svm.poly.test.err)
```

```
## Training error rate = 0
## Test error rate = 0.01
```

In this case, since I generated my data using a quadratic separation between the two classes, I would expect that a support vector machine with a polynomial kernel would outperform a support vector classifier on the training data. This expectation is confirmed by calculating the training error rate for each model, with a training error of 0 for SVM with a polynomial kernel and 0.03 for SVC. The SVM with a polynomial kernel also has a lower test error rate (0.04 vs. 0.09 for SVC).

Question 4

Here we explore the maximal margin classifier on a toy data set.

(A) We are given $n = 7$ observations in $p = 2$ dimensions. For each observation, there is an associated class label.

```r
toy.df <- tibble(Obs=seq(1,7), X1=c(3,2,4,1,2,4,4), X2=c(4,2,4,4,1,3,1),
                 Y=c("Red", "Red", "Red", "Red", "Blue", "Blue", "Blue"))
kable(toy.df)
```

Sketch the observations.

(B) Sketch the optimal separating hyperplane, and provide the equation for this hyperplane.

(C) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$, and classify to Blue otherwise." Provide the values for β_0, β_1, and β_2.

(D) On your sketch, indicate the margin for the maximal margin hyperplane.

(E) Indicate the support vectors for the maximal margin classifier.

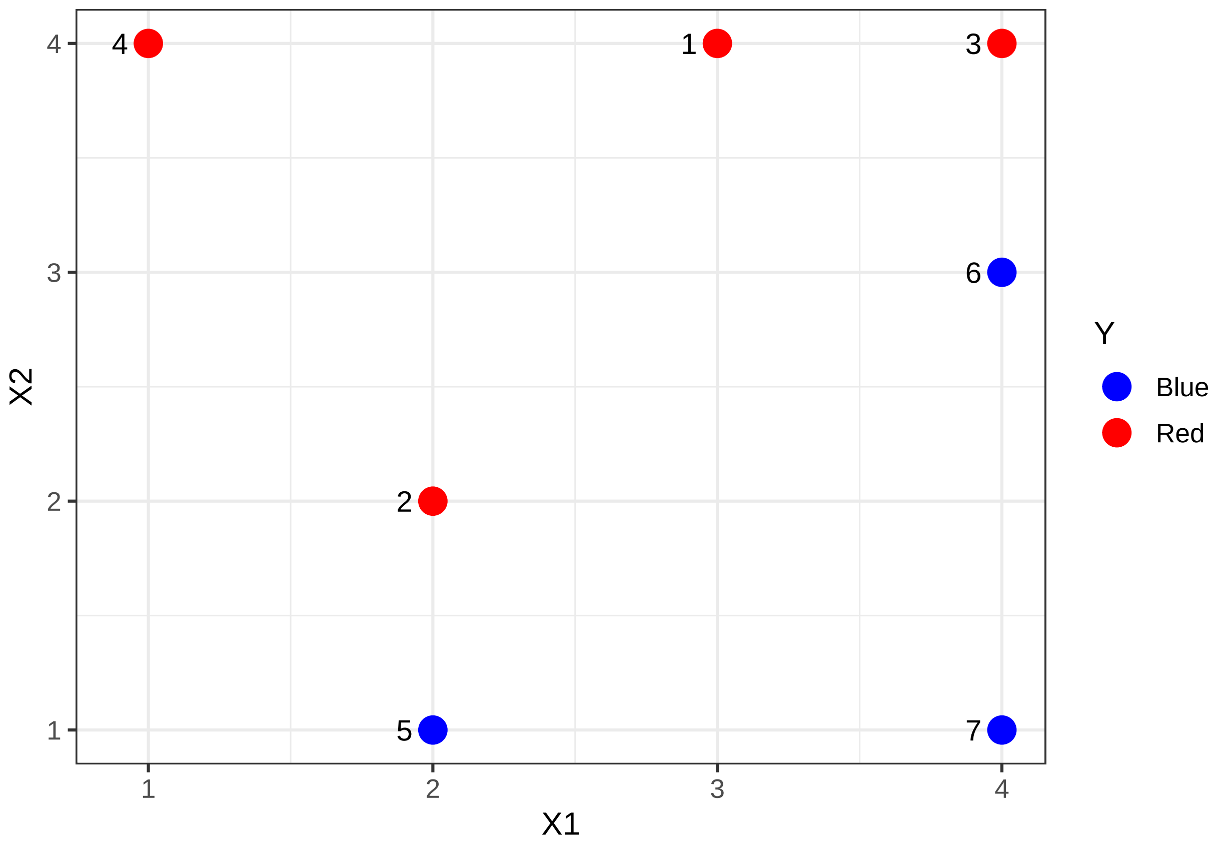
(F) Argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane.

(G) Sketch a hyperplane that is not the optimal separating hyperplane, and provide the equation for this hyperplane.

(H) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane.

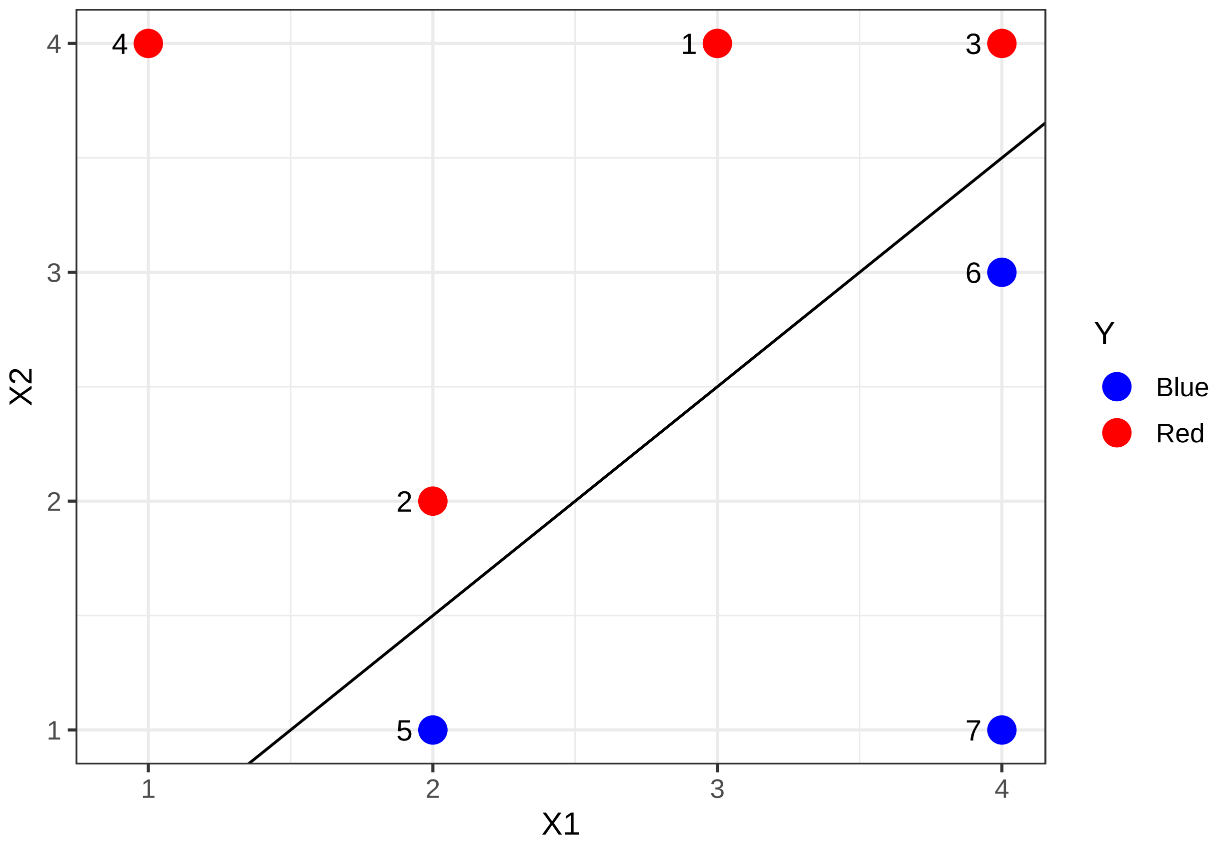
(A)

```r
toy.df <- tibble(Obs=seq(1,7), X1=c(3,2,4,1,2,4,4), X2=c(4,2,4,4,1,3,1),
                 Y=c("Red", "Red", "Red", "Red", "Blue", "Blue", "Blue"))

toy.plot <- ggplot(toy.df, aes(x=X1, y=X2, color=Y, label=Obs)) + geom_point(size=4.5) +
  geom_text(color="black", nudge_x=-0.1) + scale_color_manual(values=c("blue", "red")) +
  theme_bw(base_size=12)
toy.plot
```



(B)

```
toy.plot + geom_abline(intercept=-0.5, slope=1, color="black")
```

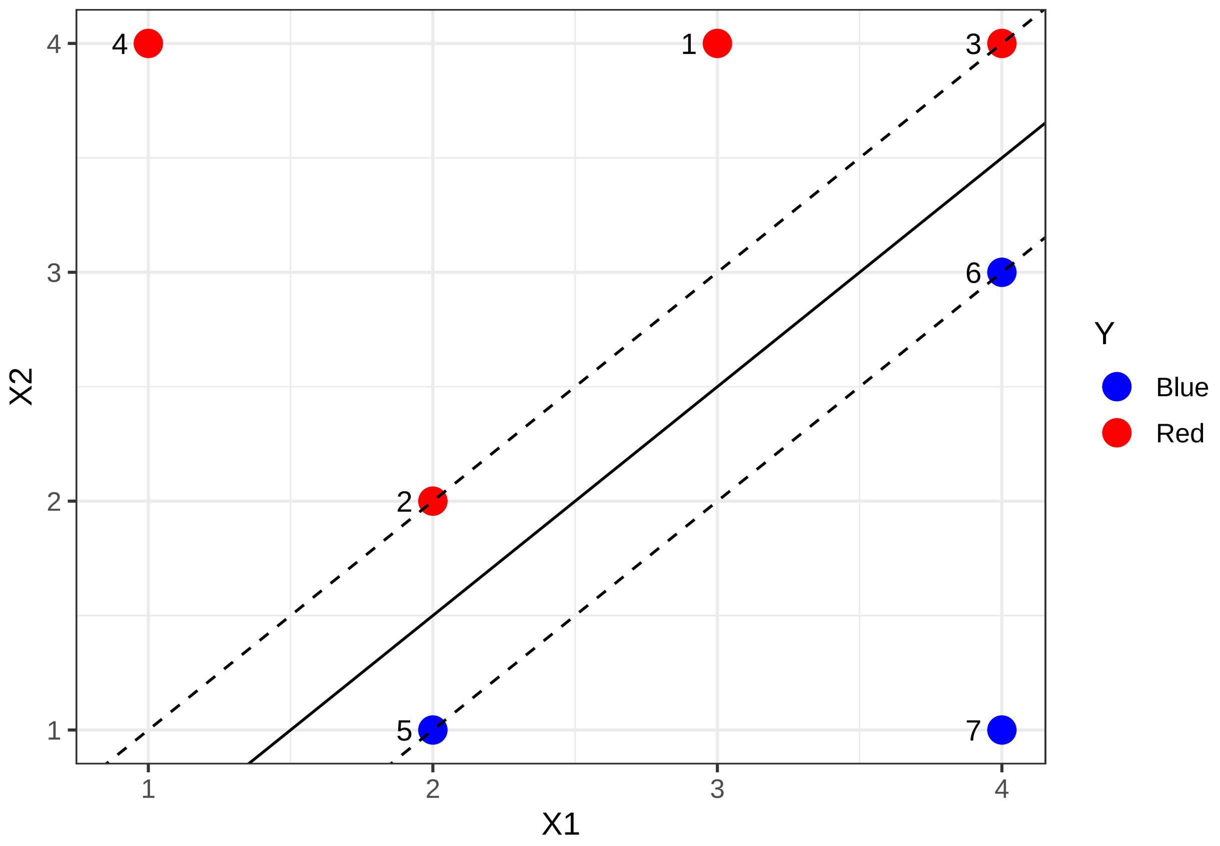


Here, I have drawn the optimal separating hyperplane in black. Its equation is: $y = x - 0.5$

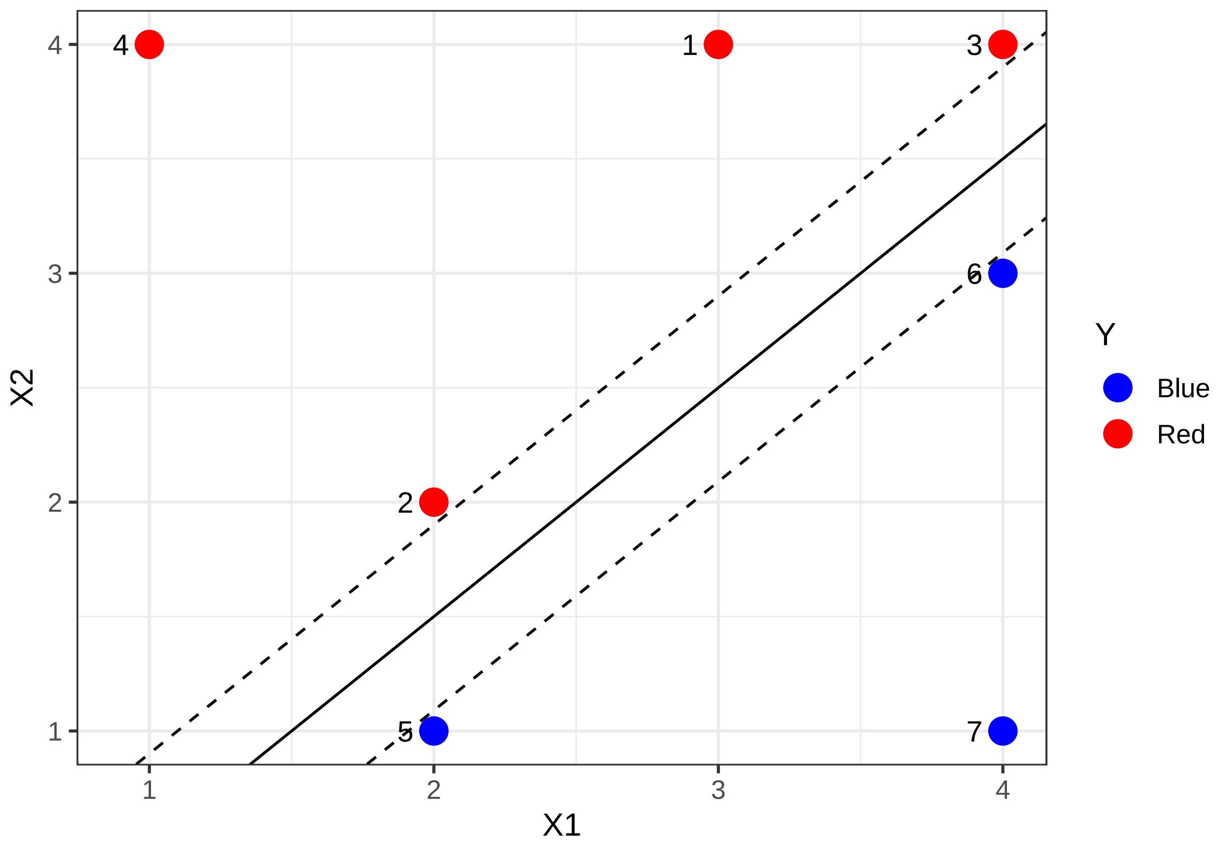
(C) The classification rule for the maximal margin classifier would be "Classify to Red if $-0.5 + x_1 - x_2 < 0$, and classify to Blue otherwise."

(D)

```r
toy.plot + geom_abline(intercept=-0.5, slope=1, color="black") +
  geom_abline(intercept=0, linetype="dashed") +
  geom_abline(intercept=-1, linetype="dashed")
```



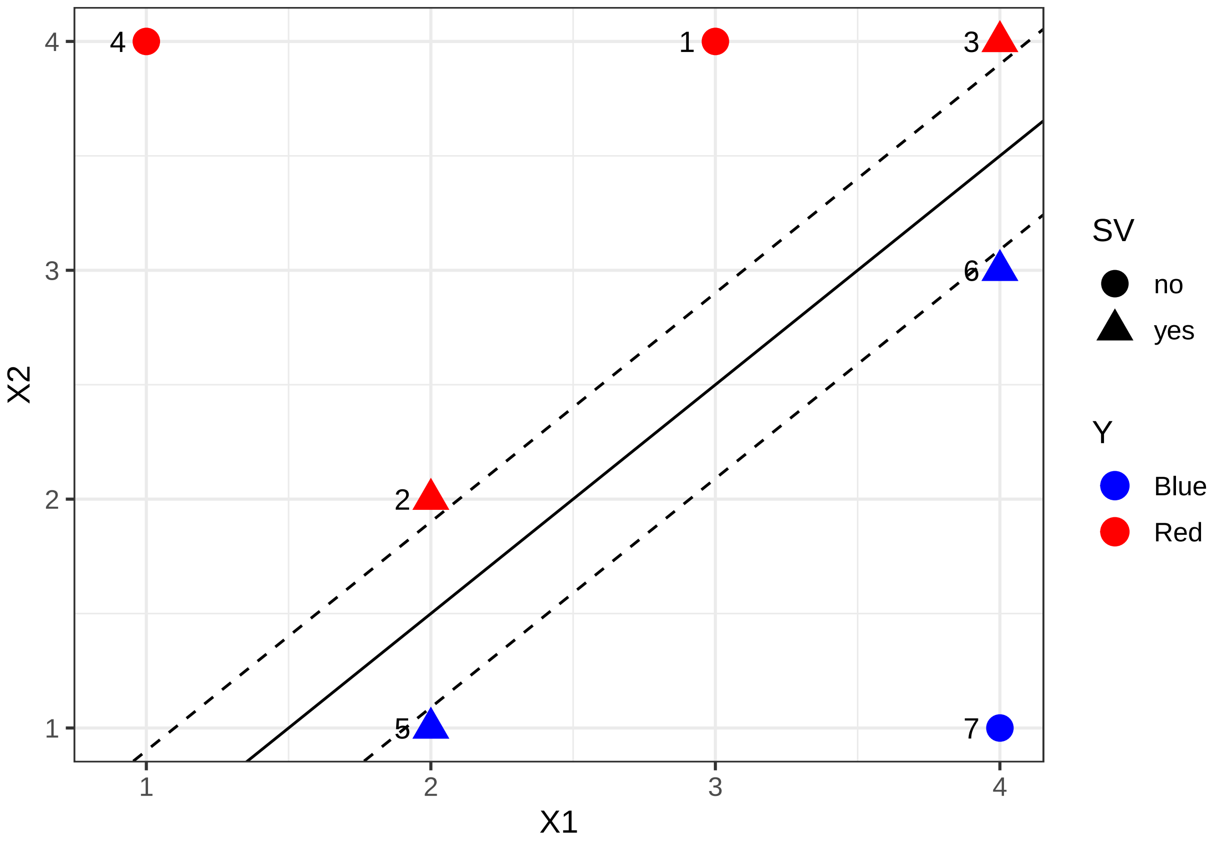
```
toy.plot + geom_abline(intercept=-0.5, slope=1, color="black") +
  geom_abline(intercept=-0.099, linetype="dashed") +
  geom_abline(intercept=-0.91, linetype="dashed")
```



The margin for the maximal margin hyperplane is shown by the dashed lines.

(E)

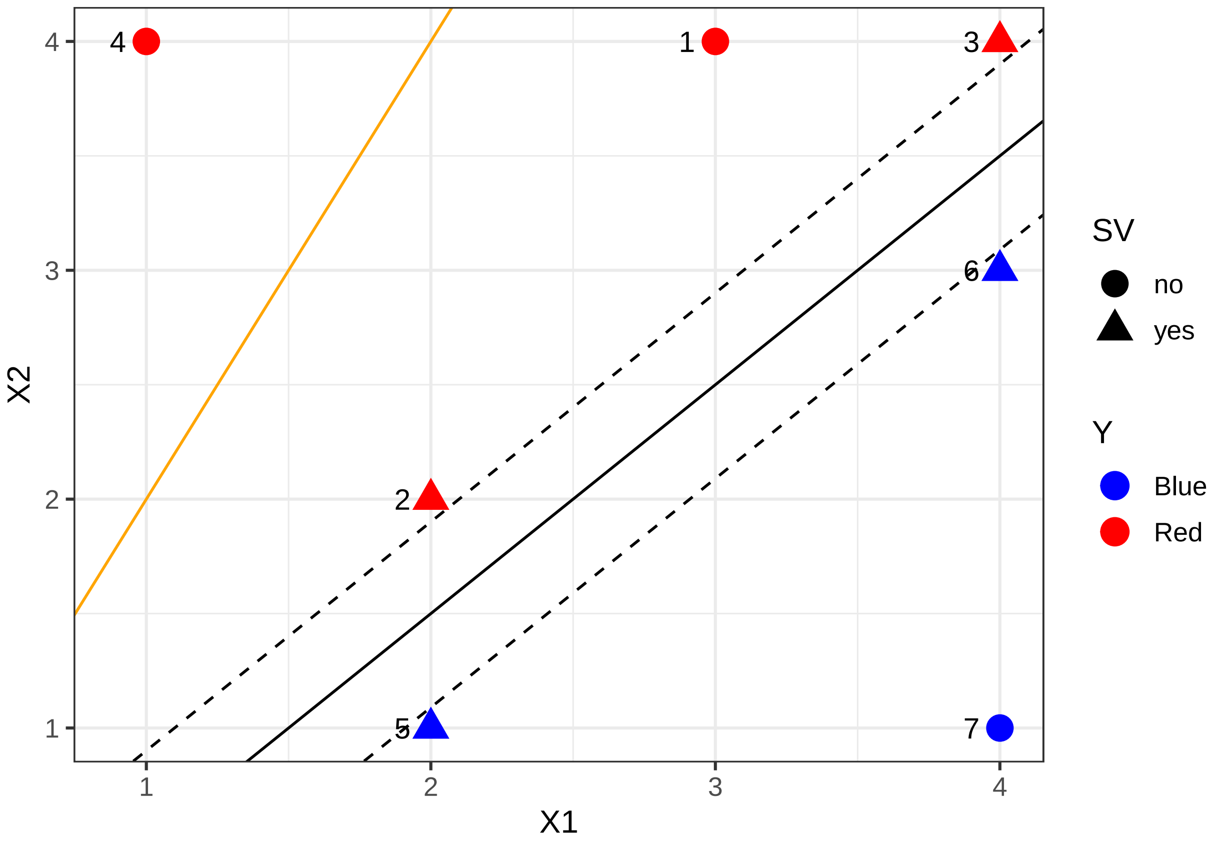
```
SV.list <- c(2,3,5,6)
toy.df <- toy.df %>% mutate(SV=ifelse(Obs %in% SV.list, "yes", "no"))

toy.plot.sv <- ggplot(toy.df, aes(x=X1, y=X2, color=Y, shape=SV, label=Obs)) + geom_point(size=4.5) +
  geom_text(color="black", nudge_x=-0.1) +
  scale_color_manual(values=c("blue", "red")) +
  theme_bw(base_size=12) + geom_abline(intercept=-0.5, slope=1, color="black") +
  geom_abline(intercept=-0.099, linetype="dashed") +
  geom_abline(intercept=-0.91, linetype="dashed")

toy.plot.sv
```



(F) Unless observation 7 were moved to to fall along or to fall past the closest margin of the maximal margin hyperplane, it should not affect the positioning of the MMH because it is not a support vector.

(G)

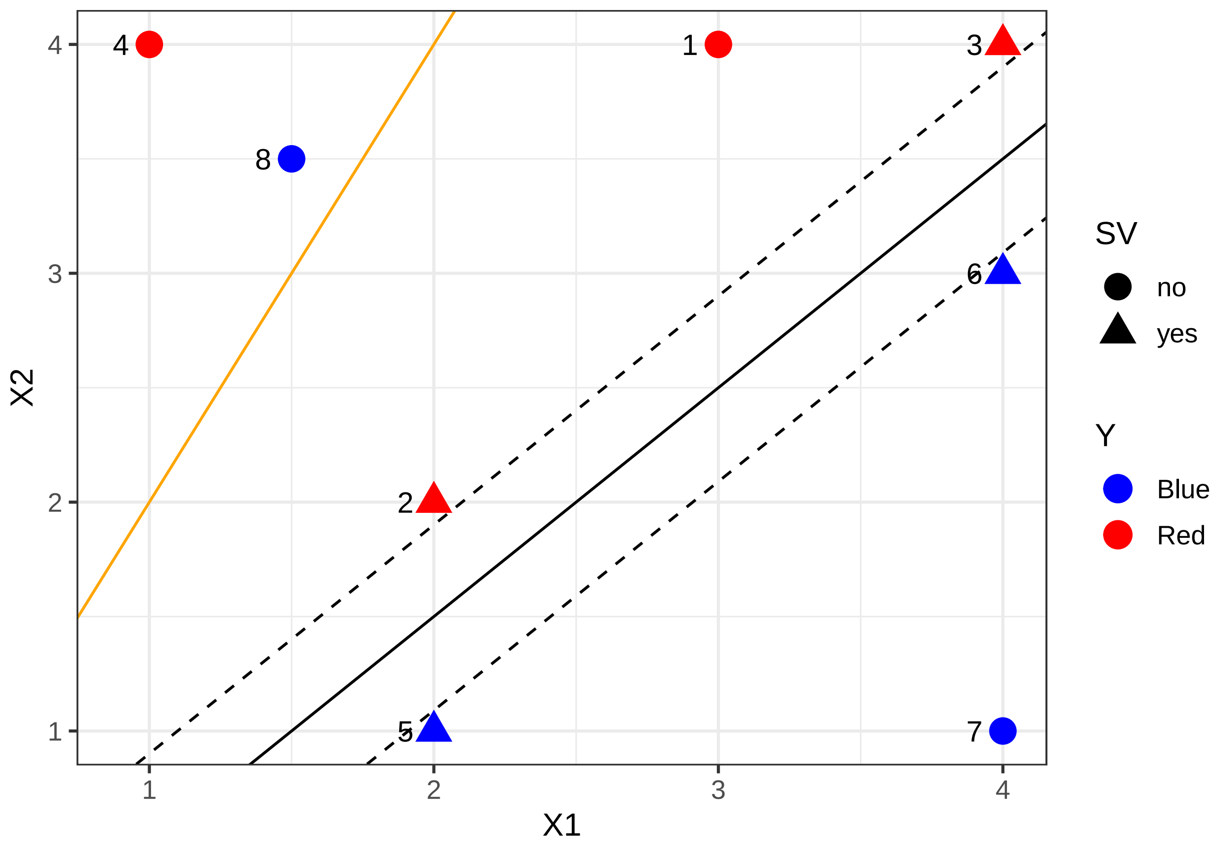
```
toy.plot.sv + geom_abline(intercept=0, slope=2, color="orange")
```



Here, I have drawn a line that is *not* the optimal separating hyperplane in orange. Its equation is $y = 2x$, or in SVM terms, $2x_1 - x_2 = 0$.

(H)

```
new.point <- tibble(Obs=8, X1=1.5, X2=3.5, Y="Blue", SV="no")
toy.df.new <- bind_rows(toy.df, new.point)

ggplot(toy.df.new, aes(x=X1, y=X2, color=Y, shape=SV, label=Obs)) + geom_point(size=4.5) +
  geom_text(color="black", nudge_x=-0.1) +
  scale_color_manual(values=c("blue", "red")) +
  theme_bw(base_size=12) + geom_abline(intercept=-0.5, slope=1, color="black") +
  geom_abline(intercept=-0.099, linetype="dashed") +
  geom_abline(intercept=-0.91, linetype="dashed") +
  geom_abline(intercept=0, slope=2, color="orange")
```



X1
X2
1
2
3
4
5
6
7
8
SV
no
yes
Y
Blue
Red