MATH 656 HW6

Jeff Gould

10/7/2020

1)

Suppose data is distributed according to the PosNeg Distribution described as follows:

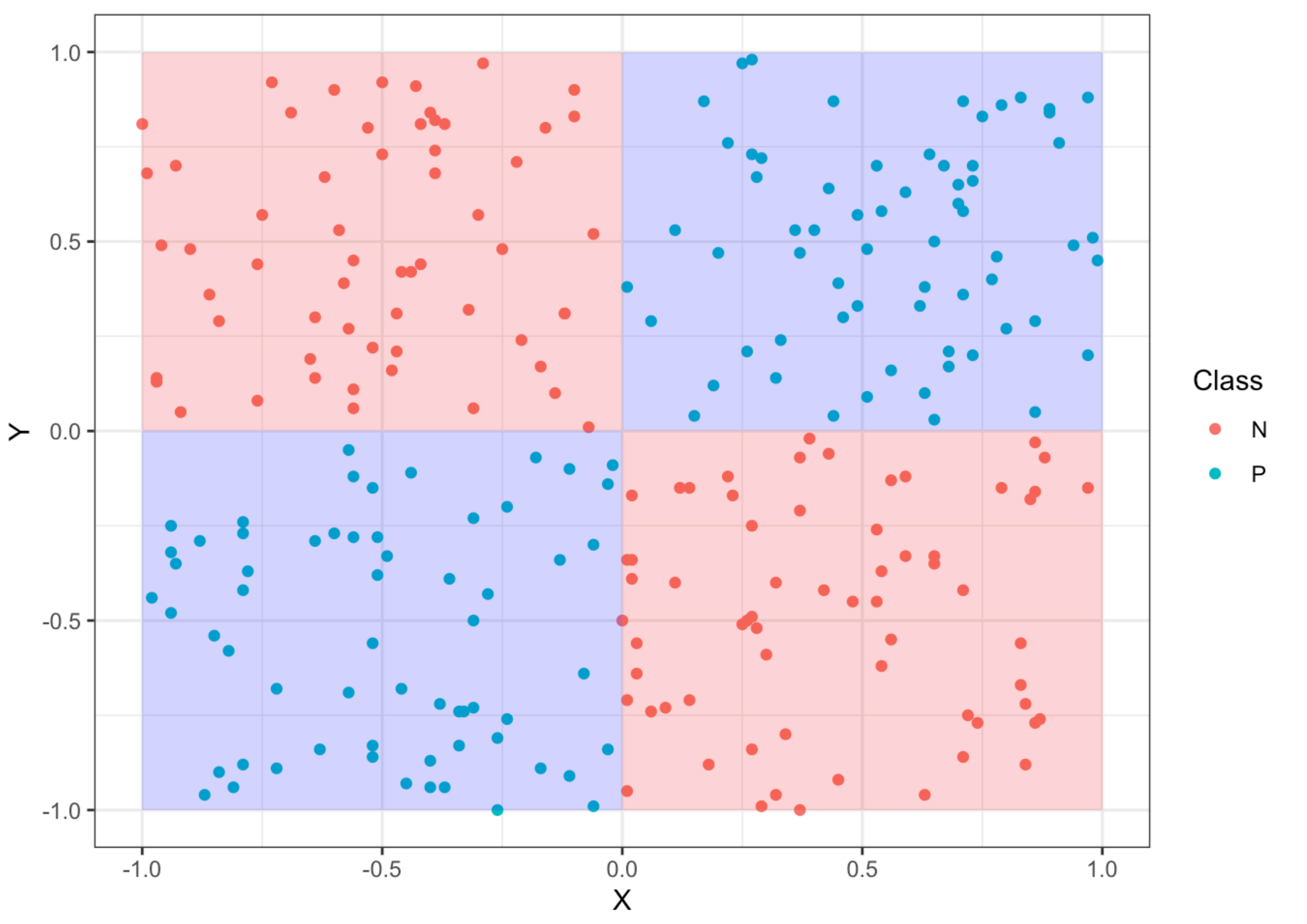
X and Y are uniform random variables on [-1,1]. If X*Y > 0, Class=P Else Class=N

You can answer the questions from the definitions of the data, but sample data is provided on Canvas in case you want to check your answers

Assume that the boundary points for binning will be chosen optimally.

```
posNeg <- read_csv("PosNeg250.csv")

ggplot() +
  geom_point(data = posNeg, aes(x = X, y = Y, color = Class)) +
  geom_rect(aes(xmin = -1, xmax = 0, ymin = -1, ymax = 0), fill = "blue", alpha = 0.2, inherit.aes = F) +
  geom_rect(aes(xmin = 0, xmax = 1, ymin = 0, ymax = 1), fill = "blue", alpha = 0.2, inherit.aes = F) +
  geom_rect(aes(xmin = -1, xmax = 0, ymin = 0, ymax = 1), fill = "red", alpha = 0.2, inherit.aes = F) +
  geom_rect(aes(xmin = 0, xmax = 1, ymin = -1, ymax = 0), fill = "red", alpha = 0.2, inherit.aes = F) +
  theme_bw()
```



a) What is the best possible rule that OneR could generate? Why?

OneR will look at each predictor, in this case X and Y, and then look at the possible classes, in this case P and N. It takes the most frequent class, and create a rule based on this class for each predictor. Then we choose which rule generated the lowest error rate. While we know the true distribution of this data is uniformly distributed over $(-1, 1) \times (-1, 1)$, which would create an equal number of P's and N's, it is possible that the training data might have a slight skew in favor of either P or N, which would influence where OneR will draw the rule. But ultimately, OneR will simply have a 50% error rate on any out of sample data.

Indeed we find that the training data has slightly more P's than N's, and they are skewed in favor of the upper-right corner of the box plotted above. So creating a OneR classification on the training data will occor based on class P.

A OneR algorithm will then divide up both X and Y in a manner to minimize the error rate, and choose whivhevr series of rules that minimizes the error rate. However, this will be an overfitting based on the randomness of the sample data, and have around a 50% error rate on testing data. Instead, it would be best to simply divide on either $X \geq 0$ or $Y \geq 0$, and classify as either P or N based on the split.

```
posNegOneR <- RWeka::OneR(formula = as.factor(Class) ~ X + Y, data = posNeg)
posNegOneR
```

```
## X:
##  < -0.89 -> N
##  < -0.77 -> P
##  < -0.525    -> N
##  < -0.505    -> P
##  < -0.385    -> N
##  < -0.255    -> P
##  < 0.14500000000000002   -> N
##  < 0.255 -> P
##  < 0.435 -> N
##  < 0.52  -> P
##  < 0.605 -> N
##  < 0.785 -> P
##  < 0.885 -> N
##  >= 0.885    -> P
## (172/250 instances correct)
```

```
summary(posNegOneR)
```

```
##
## === Summary ===
##
## Correctly Classified Instances         172               68.8    %
## Incorrectly Classified Instances        78               31.2    %
## Kappa statistic                          0.3785
## Mean absolute error                      0.312
## Root mean squared error                  0.5586
## Relative absolute error                 62.4159 %
## Root relative squared error            111.7282 %
## Total Number of Instances              250
##
## === Confusion Matrix ===
##
##    a   b   <-- classified as
##  101  22 |   a = N
##   56  71 |   b = P
```

```
testData <- expand_grid(X = seq(-1, 1, 0.01),
                        Y = seq(-1, 1, 0.01)) %>%
  mutate(Class = ifelse(X * Y >= 0, "P", "N"))

testData$predictedClass = predict(posNegOneR, testData)
mean(testData$Class == testData$predictedClass)
```

```
## [1] 0.4969184
```

b) What is the best possible rule that J48 could generate? Why?

J48 will look to minimize entropy/maximize information gain at each split. Entropy at the top node:

$-(127/250)\log_2(127/250) - (123/250)\log_2(123/250) = 0.9998$

Suppose we split on X about 0.

```
sum(posNeg$X < 0)
```

```
## [1] 121
```

$-(121/250)[(62/121)\log_2(62/121) + (59/121)\log_2(59/121)] - (129/250)[(65/129)\log_2(65/129) + (64/129)\log_2(64/129)] = 0.9997$

Split on Y about 0:

```
sum(posNeg$Y < 0)
```

```
## [1] 126
```

$-(126/250)[(62/126)\log_2(62/126) + (64/126)\log_2(64/126)] - (124/250)[(65/124)\log_2(65/124) + (59/124)\log_2(59/124)] = 0.9991$

So the first step would split on Y, as there is a marginally greater information gain, but entropy is still near a maximum, and once pruning is applied, it is best to simply not split. Instead, J48 will simplay classify all of the points as one class, which in this case is P since there are slightly more P's due to noise. But overall, the error rate is still $\approx$ 50%

```
posNegJ48 <- RWeka::J48(formula = as.factor(Class) ~ X + Y, data = posNeg)
posNegJ48
```

```
## J48 pruned tree
## ------------------
## : P (250.0/123.0)
##
## Number of Leaves  :  1
##
## Size of the tree :   1
```

```
summary(posNegJ48)
```

```
##
## === Summary ===
##
## Correctly Classified Instances         127               50.8    %
## Incorrectly Classified Instances       123               49.2    %
## Kappa statistic                          0
## Mean absolute error                      0.4999
## Root mean squared error                  0.4999
## Relative absolute error                 99.9998 %
## Root relative squared error            100      %
## Total Number of Instances              250
##
## === Confusion Matrix ===
##
##    a   b   <-- classified as
##    0 123 |   a = N
##    0 127 |   b = P
```

c) Naïve Bayes (either discrete or continuous)? Why?

For simplicity, let's use Naive Bayes in the discrete case., with X, Y either < 0 or ≥ 0

Then we get $P(Class|X, Y) = P(X|Class)P(Y|Class)P(Class)$

Say we have a point at $(0.5, 0.5)$. Then using Naive Bayes we get: $P(Class|X \geq 0, Y \geq 0) = P(X \geq 0|Class)P(Y \geq 0|Class)P(Class)$

$P(Class = P|X \geq 0, Y \geq 0) \approx P(X \geq 0|Class = P)P(Y \geq 0|Class = P)P(Class = P) = (0.5)(0.5)(0.5) = (0.5)^3$

$P(Class = N|X \geq 0, Y \geq 0) \approx P(X \geq 0|Class = N)P(Y \geq 0|Class = N)P(Class = N) = (0.5)(0.5)(0.5) = (0.5)^3$

Normalize:

$P(Class = P|X \geq 0, Y \geq 0) = \frac{(0.5)^3}{(0.5)^3+(0.5)^3} = 0.5$

Intuitively, we can see pretty easily that Naive Bayes will classify any point as either P or N with probability of 0.5, with just a little noise from the sample data.