Clasificador basado en naïve Bayes

Toma un ejemplo, imagina qué debido al clima actual, ¿se llevará a cabo un partido de cricket o no? Ahora, debemos clasificar si los jugadores jugarán el partido o no según las condiciones climáticas.

Convertir el conjunto de datos en una tabla de frecuencia

1) Crea una tabla de probabilidad encontrando las probabilidades de jugar el partido o no

2) Con base en la ecuación de Naïve Bayes, calcula la probabilidad posterior para cada clase. La probabilidad posterior más alta en cada clase es el resultado de la predicción.

3) Es fácil de usar y rápido predecir la clase de un conjunto de datos de prueba.

Funciona bien en el caso de variables de entrada categóricas en comparación con las variables numéricas.

Sus variables predictoras independientes se recomiendan para un mejor rendimiento.

Veamos, ¿cómo realizar la clasificación Naïve Bayes en R?

Cuidado si los paquetes no están instalados!

Cargar las librerías

```
> library(naivebayes)
> library(dplyr)
> library(ggplot2)
> library(psych)
```

Obtener los datos

```
> data <- read.csv("binary-Naive.csv", header = T)
> head(data)
  Launch Thickness Appearance Spreading Rank
1      0         1          9         8    2
2      0         1          8         7    1
3      0         1          7         7    1
```

4	0	1	8	9	1
5	0	1	8	7	1
6	0	1	7	7	1

Entendamos el conjunto de datos, el conjunto de datos contiene 5 columnas.
Variable Launch-Response, 0 indica producto no lanzado y 1 indica producto lanzado

Thickness-product thickness score
Appearance-product appearance score
Spreading- product spreading score
Rank-Rank of the producto

Identificación de frecuencia

Calculemos la frecuencia de la variable de respuesta debajo de cada rango. La frecuencia mínima de cada clase es 5 requerida para el análisis.

```
> xtabs(~Launch+Rank, data = data)
     Rank
Launch  1  2  3
    0 12 21 13
    1 21 15 13
```

En este caso, las frecuencias de todas las celdas son superiores a 5 y son ideales para un análisis más detallado.
Ahora solo mira cada clase de variable basada en la función str

```
> str(data)
'data.frame':   95 obs. of  5 variables:
 $ Launch    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Thickness : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Appearance: int  9 8 7 8 8 7 9 7 9 9 ...
 $ Spreading : int  8 7 7 9 7 7 8 7 9 8 ...
 $ Rank      : int  2 1 1 1 1 1 2 2 1 2 ...
```

Ahora puedes ver que el frame de datos contiene 95 (todavía un conjunto de datos pequeño, puedes probar Naive Bayes para conjuntos de datos grandes) observaciones de 5 variables.

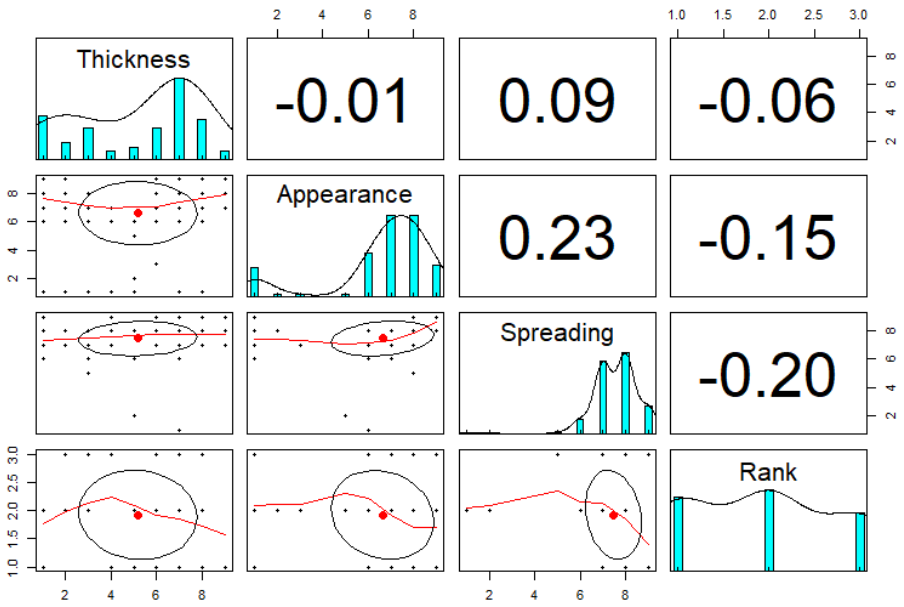
Las columnas Launch y Rank se almacenan como variables enteras. Si estas dos variables aparecen como números enteros, es necesario convertirlas en variables factoriales.

```
> data$Rank <- as.factor(data$Rank)
> data$Launch <- as.factor(data$Launch)
```

Uno de los supuestos en la clasificación de naïve Bayes es que las variables independientes no están altamente correlacionadas.
Elimina la columna de clasificación en este escenario y prueba la asociación de las variables predictoras.

Visualización

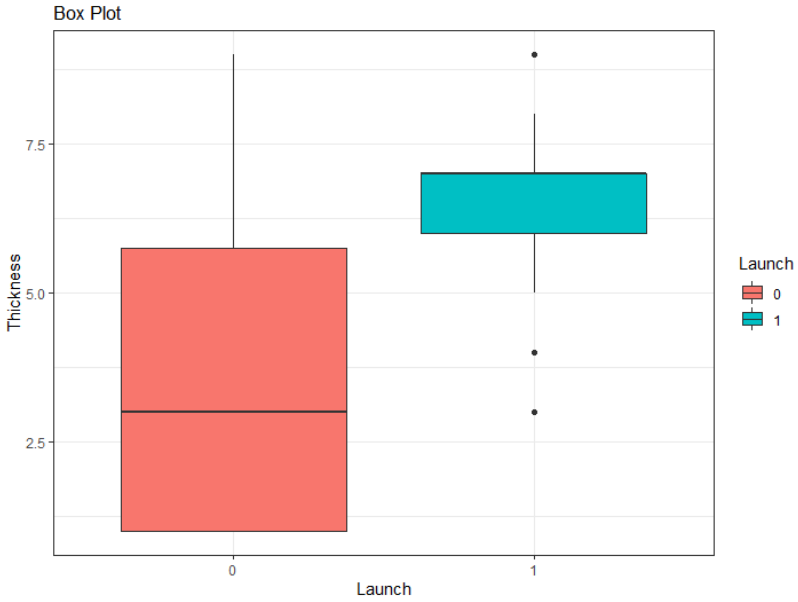
```
> pairs.panels(data[-1])
```



Se observó baja correlación entre las variables independientes.

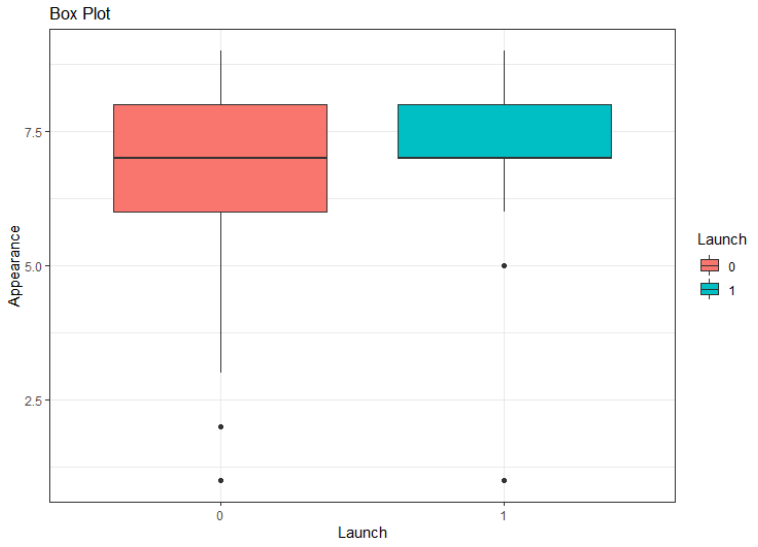
Visualizando los datos con ggplot

```
> data %>%
+ ggplot(aes(x=Launch, y=Thickness, fill = Launch)) +
+ geom_boxplot() +theme_bw()+
+ ggtitle("Box Plot")
```

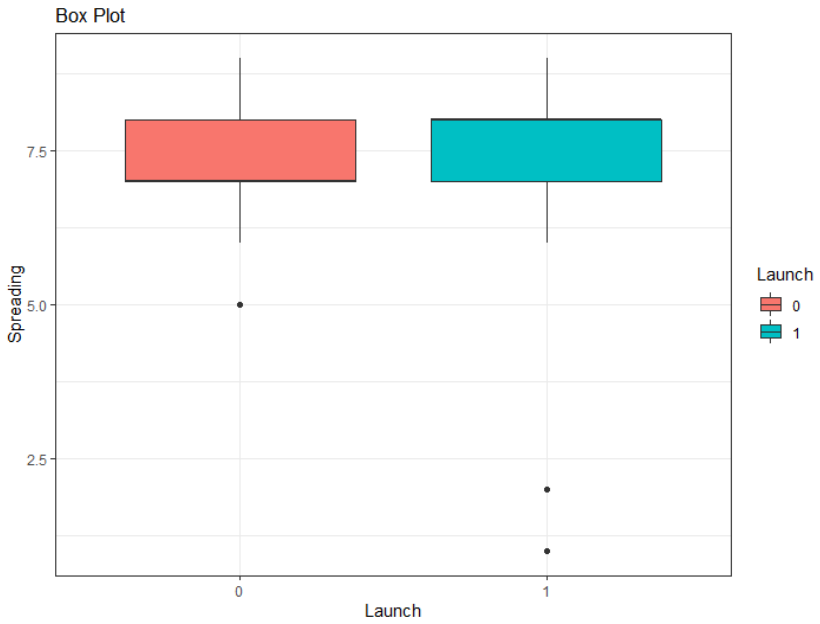


El producto obtuvo la puntuación más alta en el espesor (thickness) que se lanzó al mercado.

```
> data %>%
+ ggplot(aes(x=Launch, y=Appearance, fill = Launch)) +
+ geom_boxplot() +theme_bw()+
+ ggtitle("Box Plot")
```




```
> data %>%
+ ggplot(aes(x=Launch, y=Spreading, fill = Launch)) +
+ geom_boxplot() +theme_bw()+
+ ggtitle("Box Plot")
```



Partición de datos

Vamos a crear conjuntos de datos de entrenamiento y prueba para entrenar el modelo y probarlo.

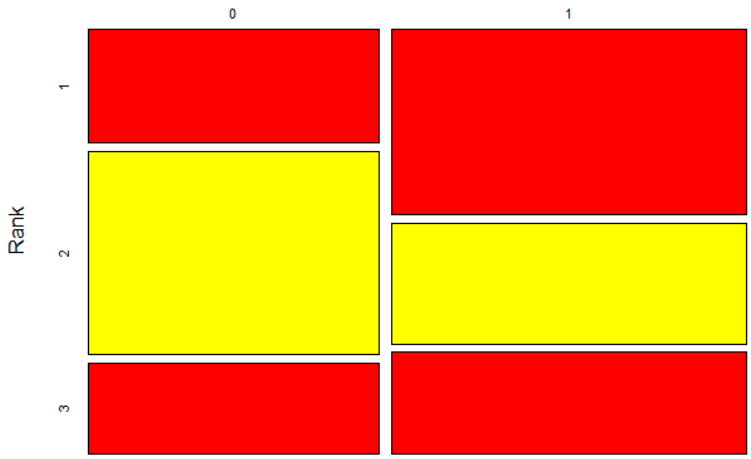
```
> set.seed(1234)
> ind <- sample(2, nrow(data), replace = T, prob = c(0.8, 0.2))
> train <- data[ind == 1,]
> test <- data[ind == 2,]
```

Clasificación naïve Bayes

Clasificación naïve Bayes en R

```
> model <- naive_bayes(Launch ~ ., data = train, usekernel = T)
> plot(model)
```

Puedes intentar usar kernel = T, según la precisión del modelo, puedes ajustar lo mismo.



Las posibilidades de lanzamiento del producto recibido en el rango 1 son muy altas y los productos recibidos en el rango 3 también tienen algunas posibilidades de un lanzamiento exitoso.

Predicción

```
> p <- predict(model, train, type = 'prob')
> head(cbind(p, train))
         0           1 Launch Thickness Appearance Spreading Rank
1 0.9987304 0.001269624      0         1          9         8    2
2 0.9928448 0.007155177      0         1          8         7    1
3 0.9974822 0.002517833      0         1          7         7    1
4 0.9884253 0.011574684      0         1          8         9    1
6 0.9974822 0.002517833      0         1          7         7    1
7 0.9987304 0.001269624      0         1          9         8    2
```

La base de la primera fila, el espesor bajo, la apariencia alta, la extensión y la puntuación de rango 2 tienen una probabilidad muy baja de lanzamiento del producto.

Matriz de confusión: datos de entrenamiento

```
> p1 <- predict(model, train)
> (tab1 <- table(p1, train$Launch))

p1  0  1
  0 28  1
  1  8 43
> 1 - sum(diag(tab1)) / sum(tab1)
[1] 0.1125
```

La clasificación errónea es de alrededor del 11%.
La precisión del modelo de entrenamiento es de alrededor del 89%, ¡no está mal!

Matriz de confusión: datos de prueba

```
> p2 <- predict(model, test)
> (tab2 <- table(p2, test$Launch))
p2  0 1
  0 6 0
  1 4 5
> 1 - sum(diag(tab2)) / sum(tab2)
[1] 0.2666667
```

Conclusión

La clasificación errónea en los datos de prueba es de aproximadamente el 26 %, según la clasificación Naïve Bayes en R.

En la prueba de entrenamiento, puedes mejorar la precisión del modelo agregando más observaciones.