Naive Bayes para Iris

para el conjunto de datos Iris, crear un clsificador Bayesiano Simple, en lenguaje R. Queremos saber cual es la probabilidad de que una observacion sea de la clase c , dado un conjunto de características o taributos x_i.

Apliacamos la fomrula de naive Bayes

$$P(c|x) = \prod_{i=1}^{n} P(x_i|c)P(c)$$

Estimación de probabilidades

Estimar $P(c)$ es tan sencillo como calcular la frecuencia de la clase en el conjunto de entrenamiento:

$$P(c) = \frac{n_c}{N}$$

Siendo n_c el numero de registros de clase c y N el total de elementos en el conjunto de entrenamiento.

cargar paquetes

En la consola de RGui de R, cargue los siguientes paquetes si aun no los ha cargado:

```
# Instalar paquetes (si no los tienes)
# el pauqte e1071 contiene la funcion naivebayes
install.packages("e1071")
# caret clasification and regresion training
install.packages("caret")
install.packages("ggplot2")

# Cargar paquetes
library(e1071)    # Contiene la función naiveBayes
library(caret)    # Para evaluación del modelo
library(ggplot2)  # Para visualización
```

Código

```
##### Cargar y explorar datos #####

# Cargar el dataset Iris (viene incluido en R)
data(iris)
```

```r
# Explorar la estructura de los datos
# str (structure), Mostrar la estructura interna de un objeto de R
str(iris)
# muestra un preview de las primeras 16 filas
head(iris)

# Proporcionar un resumen estadístico de un objeto en R
# Min. : Valor mínimo, 1st Qu.: Primer cuartil (25% de los datos están por debajo)
# Median: Mediana (valor del medio), Mean: Media (promedio)
# 3rd Qu.: Tercer cuartil (75% de los datos están por debajo)
# Max.: Valor máximo
# para variables categóricas con Species, devuelve Frecuencias de cada categoría
summary(iris)

# Verificar si hay valores missing o NA (Not Available), devuelve datos:
# Qué hace: Crea una matriz del mismo tamaño que iris con valores TRUE donde hay NA y FALSE
sum(is.na(iris))


##### Analisis exploratorio básico #####

# Distribución de las clases
# extraemos los valores de la columna Species
table(iris$Species)

# Crea una matriz de gráficos de dispersión (scatterplot matrix)
# Que Muestra todas las posibles relaciones entre pares de variables en un solo vistazo
# parámetro: [,1:4]: Selecciona todas las filas y las columnas 1 a 4
# Excluye: La columna 5 (Species) que es la variable categórición de relaciones entre varial
# col parametro que define los colors de los puntos
# iris$Species: Extrae la columna de especies
# Cómo funciona: R asigna automáticamente colores diferentes a cada categoría única
# pch "Plotting Character"  define la forma de los puntos, 16 código para punto solido
# muestra gráfico
pairs(iris[,1:4], col = iris$Species, pch = 16)

##### dividir datos entrenamiento y prueba #####

# Semilla para reproducibilidad
set.seed(123)

# la siguiente funcion pertenece al paquet caret
# iris$Species, la variable objetivo
# la función usa esta variable para estratificar la partición
# la función Mantiene la misma proporción de clases en ambos conjuntos
# p = 0 0.7 proporcion de datos que iran en el conjunto de entrenamiento
```

```r
# list = false, devuelve un vector numérico con los índices
# true devuelve una lista(util para múltiples particiones)
# Crear índices para división (70% entrenamiento, 30% prueba)
train_index <- createDataPartition(iris$Species, p = 0.7, list = FALSE)

# Crear conjuntos de entrenamiento y prueba
# iris[train_index, ], train_indes especifica los índices de las filas a seleccionar
# espacio vacio despues de la coma: significa todas las columnas
train_data <- iris[train_index, ]
# obtenemos todas as filas que no estan en train_index
test_data <- iris[-train_index, ]

# Verificar tamaños
dim(train_data)
dim(test_data)

##### Entrena el modelo #####

# la siguiente funcion pertenece al paquete e1071
# formula: Species ~ .
# ~ significa es funcion de, o es modelado por
# Species la variable que queremos predecir
# . significa todas las otras variables en el dataSet
# equivalente a: Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
# Entrenar el modelo
modelo_nb <- naiveBayes(Species ~ ., data = train_data)

# Ver el modelo entrenado
print(modelo_nb)

# Ver las probabilidades a priori
modelo_nb$apriori

# Ver las medias por clase
# tables es un componente del objeto modelo_nb que contiene las probabilidades condicionales
modelo_nb$tables

##### hacer predicciones #####

# Predecir con datos de prueba
predicciones <- predict(modelo_nb, test_data)

# Ver predicciones
# muestra un preview de las primeras 16 filas
head(predicciones)
```

```r
# Comparar con valores reales
head(test_data$Species)

##### evaluar el modelo #####

# Matriz de confusión
confusion_matrix <- table(Predicho = predicciones, Real = test_data$Species)
print(confusion_matrix)

# Métricas de evaluación
precision <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Precisión:", precision, "\n")

# Usando caret para métricas más detalladas
confusionMatrix(predicciones, test_data$Species)

##### predecir nuevas observaciones #####

# data.frame() crea una nueva tabla de datos
# Sepal.Length = c(5.1, 6.7, 5.8), Crea la columna "Sepal.Length" con 3 valores: 5.1, 6.7 y
# Crear nuevos datos para predecir
nuevos_datos <- data.frame(
  Sepal.Length = c(5.1, 6.7, 5.8),
  Sepal.Width = c(3.5, 3.0, 2.7),
  Petal.Length = c(1.4, 5.2, 4.2),
  Petal.Width = c(0.2, 2.3, 1.3)
)

# Predecir nuevas observaciones
nuevas_predicciones <- predict(modelo_nb, nuevos_datos)
print(nuevas_predicciones)

# También obtener probabilidades
probabilidades <- predict(modelo_nb, nuevos_datos, type = "raw")
print(probabilidades)
```

Salidas

Salida esperada de: modelo_nb$tables:

```
modelo_nb$tables

# Output esperado:
$Sepal.Length
            Sepal.Length
Y                [,1]      [,2]
```

```
  setosa     5.0060000 0.3524897
  versicolor 5.9360000 0.5161711
  virginica  6.5880000 0.6358796

$Sepal.Width
            Sepal.Width
Y                [,1]      [,2]
  setosa     3.4280000 0.3790644
  versicolor 2.7700000 0.3137983
  virginica  2.9740000 0.3224966

# ... y similar para Petal.Length y Petal.Width
```

Interpretación de los valores: Para cada variable y clase, muestra dos valores:

Para Sepal.Length y setosa: Media = 5.006, Desviación estándar = 0.352

Esto significa que: * La longitud promedio del sépalo para setosa es 5.006 cm * La variabilidad alrededor de esa media es ± 0.352 cm