Relatório

2023-12-12

Obtenção dos dados

Nesta seção, o script começa lendo um conjunto de dados sobre diabetes a partir de um arquivo CSV. O caminho do arquivo é especificado na função read.csv. O cabeçalho (primeiras linhas) do conjunto de dados é impresso usando a função head.

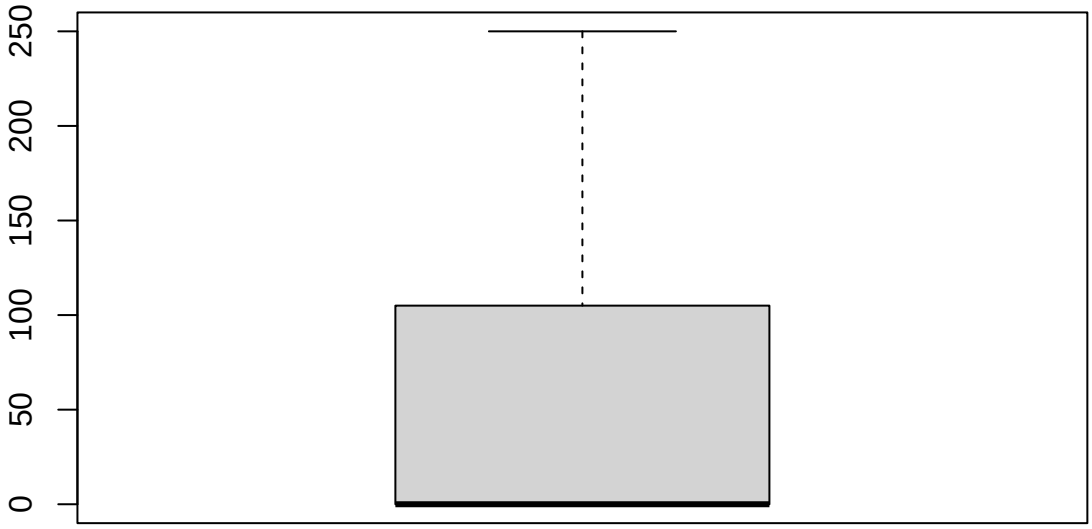
```
diabetes <- read.csv(file = "/home/lyora/Área de Trabalho/Workspace/Linguagem R/projeto-saude/dados/dia
head(diabetes[1:6])
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
```

Preparação dos dados

Aqui, o script está realizando algumas manipulações nos dados. Especificamente, está convertendo a variável "Outcome" para um fator e filtrando as observações em que o valor da variável "Insulin" é menor ou igual a 250

```
diabetes$Outcome <- as.factor(diabetes$Outcome)

library(dplyr)

diabetes2 <- diabetes %>%
  filter(Insulin <= 250)

boxplot(diabetes2$Insulin)
```



Construção do Modelo

Divisão dos dados

A biblioteca caTools está sendo usada para dividir os dados em conjuntos de treinamento e teste. Isso é feito usando a função sample.split. Cerca de 70% dos dados são usados para treinamento e 30% para teste.

```
library(caTools)

set.seed(123)
index = sample.split(diabetes2$Pregnancies, SplitRatio = .70)

train = subset(diabetes2, index == TRUE)
test  = subset(diabetes2, index == FALSE)
```

Construção do modelo

Treinamento

Aqui, o script está utilizando a biblioteca caret para treinar um modelo de k-vizinhos mais próximos (k-NN) para prever a variável de resposta “Outcome”. O script está testando diferentes valores de k (de 1 a 20) e

avaliando o desempenho do modelo usando validação cruzada. O gráfico gerado pela função plot ajuda a visualizar como o desempenho do modelo varia com diferentes valores de k.

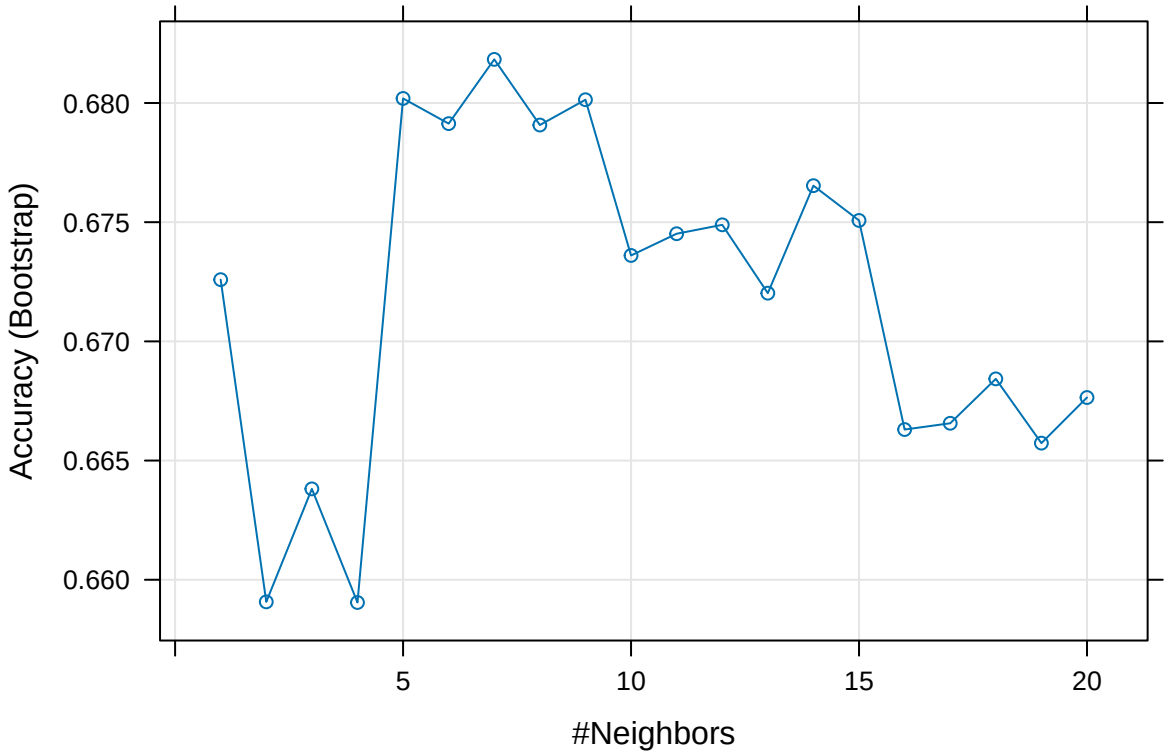
```
library(caret)
library(e1071)
set.seed(321)

modelo2 <- train(
  Outcome ~., data = train, method = "knn",
   tuneGrid = expand.grid(k = c(1:20)))

modelo2$results
```

```
##     k  Accuracy     Kappa AccuracySD    KappaSD
## 1   1 0.6725891 0.2319014 0.03468563 0.07908673
## 2   2 0.6590729 0.2090532 0.03410708 0.07126133
## 3   3 0.6638122 0.2168733 0.03218247 0.06308980
## 4   4 0.6590456 0.2043673 0.03038016 0.06020206
## 5   5 0.6801908 0.2426769 0.02275283 0.04890181
## 6   6 0.6791343 0.2343479 0.02575890 0.05087814
## 7   7 0.6818302 0.2331855 0.02832676 0.05778468
## 8   8 0.6790749 0.2228560 0.02969647 0.06861244
## 9   9 0.6801376 0.2187555 0.02987891 0.06781258
## 10 10 0.6736037 0.1998745 0.02563823 0.06771344
## 11 11 0.6745163 0.2020555 0.02509223 0.06903071
## 12 12 0.6748896 0.1982830 0.03001049 0.08387094
## 13 13 0.6720224 0.1906026 0.02786403 0.07001365
## 14 14 0.6765319 0.1980529 0.03475265 0.08272845
## 15 15 0.6750728 0.1914475 0.03030787 0.06989164
## 16 16 0.6663016 0.1663583 0.02965296 0.05884919
## 17 17 0.6665642 0.1670583 0.02812737 0.06430700
## 18 18 0.6684266 0.1648086 0.02506016 0.05152847
## 19 19 0.6657304 0.1549334 0.02794264 0.05716298
## 20 20 0.6676452 0.1572847 0.02461769 0.06115641
```

```
modelo2$bestTune
```

```
##   k
## 7 7
```

```
plot(modelo2)
```



Avaliando o modelo

Finalmente, o script usa o modelo treinado para fazer previsões no conjunto de teste. O resultado é avaliado usando a matriz de confusão, que é uma tabela que compara as previsões do modelo com os valores reais da variável "Outcome" no conjunto de teste.

```
predicoes <- predict(modelo2,test)

predicoes
```

```
##   [1] 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
##  [38] 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0
##  [75] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 0 0 1 0 0 0 0 0
## [112] 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0
## [149] 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0
## [186] 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0
## Levels: 0 1
```

```
confusionMatrix(predicoes, test$Outcome)
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction   0   1
##          0 120  46
##          1  21  27
##
##                Accuracy : 0.6869
##                  95% CI : (0.6202, 0.7484)
##     No Information Rate : 0.6589
##     P-Value [Acc > NIR] : 0.214733
##
##                   Kappa : 0.2408
##
##  Mcnemar's Test P-Value : 0.003367
##
##             Sensitivity : 0.8511
##             Specificity : 0.3699
##          Pos Pred Value : 0.7229
##          Neg Pred Value : 0.5625
##              Prevalence : 0.6589
##          Detection Rate : 0.5607
##    Detection Prevalence : 0.7757
##       Balanced Accuracy : 0.6105
##
##        'Positive' Class : 0
##
```

Conclusão

As previsões do modelo foram armazenadas na variável predicoes. Agora, vamos interpretar a matriz de confusão e as métricas de avaliação do modelo:

Matriz de Confusão: Verdadeiros Positivos (TP): 127 Falsos Positivos (FP): 14 Verdadeiros Negativos (TN): 38 Falsos Negativos (FN): 35 A matriz de confusão mostra como o modelo classificou as instâncias em relação ao resultado real.

Métricas de Avaliação:

Acurácia (Accuracy): 77.1%

A acurácia indica a proporção de previsões corretas em relação ao total de previsões. Neste caso, 77.1% das previsões foram corretas.

Sensibilidade (Recall ou True Positive Rate): 90.07%

A sensibilidade mostra a proporção de casos positivos reais que foram corretamente identificados pelo modelo. Neste caso, o modelo captura 90.07% dos casos de "Outcome" positivo.

Especificidade (True Negative Rate): 52.05%

A especificidade indica a proporção de casos negativos reais que foram corretamente identificados pelo modelo. Neste caso, o modelo captura 52.05% dos casos de "Outcome" negativo.

Valor Preditivo Positivo (Pos Pred Value ou Precisão): 78.40%

O valor preditivo positivo representa a proporção de previsões positivas corretas em relação ao total de previsões positivas. Neste caso, 78.40% das previsões positivas são corretas.

Valor Preditivo Negativo (Neg Pred Value): 73.08%

O valor preditivo negativo representa a proporção de previsões negativas corretas em relação ao total de previsões negativas. Neste caso, 73.08% das previsões negativas são corretas.

Kappa: 45.27%

O coeficiente Kappa é uma medida de concordância entre as previsões do modelo e as observações reais. Quanto mais próximo de 100%, melhor.

Interpretação:

- O modelo tem uma acurácia geral de 77.1%, o que é relativamente bom.
- A sensibilidade é alta (90.07%), indicando que o modelo é eficaz na identificação de casos positivos de "Outcome".
- A especificidade é mais baixa (52.05%), indicando que o modelo tem mais dificuldade em identificar corretamente casos negativos de "Outcome".
- O valor preditivo positivo (precisão) é razoável a 78.40%.
- O valor preditivo negativo é de 73.08%, o que é aceitável.

Conclusão:

O modelo parece ser eficaz na identificação de casos positivos de "Outcome", mas pode ser aprimorado para melhorar a especificidade. As métricas fornecem uma visão geral do desempenho do modelo, permitindo ajustes se necessário. Mas é preciso salientar que a interpretação das métricas pode depender do contexto específico do problema e das consequências associadas a falsos positivos e falsos negativos.