 # Machine Learning - Classificação

1. Working Directory

Configurando o diretório de trabalho

setwd("C:/Users/Utilizador/repos/Formacao_cientista_de_dados/big_data_analytics_R_microsoft_azure_machine_lear
getwd()

2. Bussines Problems

Previsão de Ocorrência de Câncer de Mama

Os dados do câncer da mama incluem 569 observações de biópsias de câncer, cada um com 32 características (variáveis).

Uma característica é um número de identificação (ID), outro é o diagnóstico de câncer, e 30 são medidas laboratoriais numéricas.

O diagnóstico é codificado como "M" para indicar maligno ou "B" para indicar benigno.

http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

3. Data Loading dados <- read.csv("dataset.csv", stringsAsFactors = FALSE)

Visualizando as variáveis (resumo)

str(dados)

Visualizando os dados

View(dados)

4. Pre-Processing

Pré processamento

Excluindo a coluna ID

Independentemente do método de aprendizagem de máquina, deve sempre ser excluídas variáveis de ID.

Caso contrário, isso pode levar a resultados errados porque o ID pode ser usado para unicamente

"prever" cada exemplo.

Por conseguinte, um modelo que inclui um identificador pode sofrer de superajuste (overfitting),

e será muito difícil usá-lo para generalizar outros dados.

dados$id = NULL

Ajustando o label da variável alvo: trocando o nome das linhas

dados$diagnosis = sapply(dados$diagnosis, function(x){ifelse(x=='M', 'Maligno', 'Benigno')})

Muitos classificadores requerem que as variáveis sejam do tipo Fator

Criar tabela de contingência

table(dados$diagnosis)

Convertendo a coluna diagnosis em factor

dados$diagnosis <- factor(dados$diagnosis, levels = c("Benigno", "Maligno"), labels = c("Benigno", "Maligno"))

Visualizando as váriaveis (resumo)

str(dados$diagnosis)

Verificando a proporção (Porcetagem)

round(prop.table(table(dados$diagnosis)) * 100, digits = 1)

5. Descriptive statistics

Descreve, compreende, organiza e resumi os dados

Medidas de Tendência Central

Detectamos um problema de escala entre os dados, que então precisam ser normalizados.

Alguns dados estão com ponto outros com vírgula

O cálculo de distância feito pelo kNN é dependente das medidas de escala nos dados de entrada.

summary(dados[c("radius_mean", "area_mean", "smoothness_mean")])

5.1 Normalization

Criando um função de normalização

normalizar <- function(x) { return ((x - min(x)) / (max(x) - min(x))) }

Testando a função de normalização - os resultados devem ser idênticos

normalizar(c(1, 2, 3, 4, 5)) normalizar(c(10, 20, 30, 40, 50))

5.2 Applying Normalization dados_norm <- as.data.frame(lapply(dados[2:31], normalizar))

Visualização dos dados

View(dados_norm)

6. Training the Model

Treinando o modelo com KNN

Imports

Carregando o pacote library

install.packages("class") library(class) ?knn

6.1 Test and Training Data

Criando dados de treino

dados_treino <- dados_norm[1:469,]

Criando dados de teste

dados_teste <- dados_norm[470:569,]

Criando os labels para os dados de treino e de teste

dados_treino_labels <- dados[1:469, 1] dados_teste_labels <- dados[470:569, 1] length(dados_treino_labels) length(dados_teste_labels)

6.2 Model

Criando o modelo

modelo_knn_v1 <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k = 21)

A função knn() retorna um objeto do tipo fator com as previsões para cada exemplo no dataset de teste

summary(modelo_knn_v1)

7. Evaluating the Model's Performance

Avaliando a Performance do Modelo

Carregando o gmodels

library(gmodels)

Criando uma tabela cruzada dos dados previstos x dados atuais

Usaremos amostra com 100 observações: length(dados_teste_labels)

CrossTable(x = dados_teste_labels, y = modelo_knn_v1, prop.chisq = FALSE)

Interpretando os Resultados

A tabela cruzada mostra 4 possíveis valores, que representam os falso/verdadeiro positivo e negativo

Temos duas colunas listando os labels originais nos dados observados

Temos duas linhas listando os labels dos dados de teste

Temos:

Cenário 1: Célula Benigno (Observado) x Benigno (Previsto) - 61 casos - true positive

Cenário 2: Célula Maligno (Observado) x Benigno (Previsto) - 00 casos - false positive (o modelo errou)

Cenário 3: Célula Benigno (Observado) x Maligno (Previsto) - 02 casos - false negative (o modelo errou)

Cenário 4: Célula Maligno (Observado) x Maligno (Previsto) - 37 casos - true negative

Lendo a Confusion Matrix (Perspectiva de ter ou não a doença):

True Negative = nosso modelo previu que a pessoa NÃO tinha a doença e os dados mostraram que realmente a pessoa NÃO tinha a doença

False Positive = nosso modelo previu que a pessoa tinha a doença e os dados mostraram que NÃO, a pessoa tinha a doença

False Negative = nosso modelo previu que a pessoa NÃO tinha a doença e os dados mostraram que SIM, a pessoa tinha a doença

True Positive = nosso modelo previu que a pessoa tinha a doença e os dados mostraram que SIM, a pessoa tinha a doença

Falso Positivo - Erro Tipo I

Falso Negativo - Erro Tipo II

Otimizando a Performance do Modelo

Usando a função scale() para padronizar o z-score

?scale() dados_z <- as.data.frame(scale(dados[-1]))

Confirmando transformação realizada com sucesso

summary(dados_z$area_mean)

Criando novos datasets de treino e de teste

dados_treino <- dados_z[1:469,] dados_teste <- dados_z[470:569,]

dados_treino_labels <- dados[1: 469, 1] dados_teste_labels <- dados[470: 569, 1]

Reclassificando

modelo_knn_v2 <- knn(train = dados_treino, test = dados_teste, cl = dados_treino_labels, k = 21)

Criando uma tabela cruzada dos dados previstos x dados atuais

CrossTable(x = dados_teste_labels, y = modelo_knn_v2, prop.chisq = FALSE)

Experimente diferentes valores para k

9. Building a Model with Support Vector Machine (SVM)

Etapa 6: Construindo um Modelo com Algoritmo Support Vector Machine (SVM)

Definindo a semente para resultados reproduzíveis

set.seed(40)

Prepara o dataset

dados <- read.csv("dataset.csv", stringsAsFactors = TRUE)

dados$id = NULL

dados[,'index'] <- ifelse(runif(nrow(dados)) < 0.8,1,0)

View(dados)

Dados de treino e teste

trainset <- dados[dados$index == 1,] testset <- dados[dados$index==0,]

Obter o índice

trainColNum <- grep('index', names(trainset))

Remover o índice dos datasets

trainset <- trainset[,-trainColNum] testset <- testset[,-trainColNum]

Obter índice de coluna da variável target no conjunto de dados

typeColNum <- grep('diag',names(dados))

Cria o modelo

Nós ajustamos o kernel para radial, já que este conjunto de dados não tem um

plano linear que pode ser desenhado

library(e1071) ?svm modelo_svm_v1 <- svm(diagnosis ~ ., data = trainset, type = 'C-classification', kernel = 'radial')

10. Prevision

Previsões nos dados de treino

pred_train <- predict(modelo_svm_v1, trainset)

Percentual de previsões corretas com dataset de treino

mean(pred_train == trainset$diagnosis)

Previsões nos dados de teste

pred_test <- predict(modelo_svm_v1, testset)

Percentual de previsões corretas com dataset de teste

mean(pred_test == testset$diagnosis)

Confusion Matrix

table(pred_test, testset$diagnosis)

11. Random Forest

Construindo um Modelo com Algoritmo Random Forest

Criando o modelo

library(rpart) modelo_rf_v1 = rpart(diagnosis ~ ., data = trainset, control = rpart.control(cp = .0005))

Previsões nos dados de teste

tree_pred = predict(modelo_rf_v1, testset, type='class')

Percentual de previsões corretas com dataset de teste

mean(tree_pred==testset$diagnosis)

Confusion Matrix

table(tree_pred, testset$diagnosis)