UNIVERSIDADE FEDERAL DO PARANÁ
Especialização em Inteligência Artificial Aplicada
IAA004 - Linguagem R
Prof. Dr. Razer A N R Montaño

TRABALHO DA DISCIPLINA

Aluna: Luiza Ruivo Marinho Santos

1 Pesquisa com Dados de Satélite (Satellite)

O banco de dados consiste nos valores multi-espectrais de pixels em vizinhanças 3x3 em uma imagem de satélite, e na classificação associada ao pixel central em cada vizinhança. O objetivo é prever esta classificação, dados os valores multi-espectrais.

Um quadro de imagens do Satélite Landsat com MSS (Multispectral Scanner System) consiste em quatro imagens digitais da mesma cena em diferentes bandas espectrais. Duas delas estão na região visível (correspondendo aproximadamente às regiões verde e vermelha do espectro visível) e duas no infravermelho (próximo). Cada pixel é uma palavra binária de 8 bits, com 0 correspondendo a preto e 255 a branco. A resolução espacial de um pixel é de cerca de 80m x 80m. Cada imagem contém 2340 x 3380 desses pixels. O banco de dados é uma subárea (minúscula) de uma cena, consistindo de 82 x 100 pixels. Cada linha de dados corresponde a uma vizinhança quadrada de pixels 3x3 completamente contida dentro da subárea 82x100. Cada linha contém os valores de pixel nas quatro bandas espectrais (convertidas em ASCII) de cada um dos 9 pixels na vizinhança de 3x3 e um número indicando o rótulo de classificação do pixel central.

As classes são: solo vermelho, colheita de algodão, solo cinza, solo cinza úmido, restolho de vegetação, solo cinza muito úmido.

Os dados estão em ordem aleatória e certas linhas de dados foram removidas, portanto você não pode reconstruir a imagem original desse conjunto de dados. Em cada linha de dados, os quatro valores espectrais para o pixel superior esquerdo são dados primeiro, seguidos pelos quatro valores espectrais para o pixel superior central e, em seguida, para o pixel superior direito, e assim por diante, com os pixels lidos em sequência, da esquerda para a direita e de cima para baixo. Assim, os quatro valores espectrais para o pixel central são dados pelos atributos 17, 18, 19 e 20. Se você quiser, pode usar apenas esses quatro atributos, ignorando os outros. Isso evita o problema que surge quando uma vizinhança 3x3 atravessa um limite.

O banco de dados se encontra no pacote **mlbench** e é completo (não possui dados faltantes).

O arquivo `1_comandos.R` possui todos os comandos listados da atividade.

Instalar o pacote do modelo de treinamento *RandomForest*:

```
install.packages("randomForest")
library(randomForest)
```

Instalar o pacote *mlbench* para obter o banco de dados:

```
install.packages("mlbench")
library(mlbench)
```

Instalar pacote *caret* para usar a função `createDataPartition()`:

```
install.packages("caret")
library("caret")
```

Instalar o pacote *e1071* para o treinamento dos modelos:

```
install.packages("e1071")
library("e1071")
```

Carregar o dataset com o banco de dados Satellite:

```
data(Satellite)
dataset <- Satellite
```

Criar bases de treino e teste e particiona a bases em treino (80%) e teste (20%):

```
indexes <- createDataPartition(dataset$classes, p=0.80,
                               list=FALSE)
training <- dataset[indexes,]
test <- dataset[-indexes,]
```

Tarefas:

1. **Treine modelos RandomForest, SVM e RNA para predição destes dados.**

```
set.seed(0)

rf <- train(classes~., data=training, method="rf") # RandomForest
svm <- train(classes~., data=training, method="svmRadial") # SVM
rna <- train(classes~., data=training, method="nnet",
         trace=FALSE) # RNA
```

Aplicar modelos treinados na base de teste:

```
predict.rf <- predict(rf, test)
predict.svm <- predict(svm, test)
predict.rna <- predict(rna, test)
```

Verificar a quantidade de amostras de cada classe na base de teste:

```
frequency <- table(dataset$classes)
frequency_rel <- prop.table(frequency)
data.frame('Classe'= names(frequency),
           'Frequência'= as.vector(frequency),
           'Relativa'= as.vector(frequency_rel),
           'Porcentagem'= as.vector(frequency_rel) * 100)
```

Resultado:

```
                Classe   Frequência    Relativa    Porcentagem
1            red soil       1533     0.2382284       23.82284
2         cotton crop        703     0.1092463       10.92463
3           grey soil       1358     0.2110334       21.10334
4      damp grey soil        626     0.0972805        9.72805
5  vegetation stubble        707     0.1098679       10.98679
6 very damp grey soil       1508     0.2343434       23.43434
```

2. **Escolha o melhor modelo com base em suas matrizes de confusão.**

RandomForest:

```
confusionMatrix(predict.rf, test$classes)
```

Accuracy: 0.9245

SVM:

```
confusionMatrix(predict.svm, test$classes)
```

Accuracy: 0.9112

RNA:

```
confusionMatrix(predict.rna, test$classes)
```

Accuracy: 0.7874

Levando em consideração a acurácia, o modelo **RandomForest** é o que melhor se saiu bem.

3. **Treine o modelo final com todos os dados e faça a predição na base completa.**

```
 print(rf)
```

Resultado:

```
Random Forest

5151 samples
  36 predictor
   6 classes: 'red soil', 'cotton crop', 'grey soil', 'damp grey soil',
'vegetation stubble', 'very damp grey soil'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 5151, 5151, 5151, 5151, 5151, 5151, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.9054192  0.8827752
  19    0.9054027  0.8828494
  36    0.8976296  0.8732268

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Treina o modelo final:

```
final_model <- randomForest(classes~., data=dataset, mtry=2,
importance=TRUE)

final_predict <- predict(final_model, dataset)

confusionMatrix(final_predict, dataset$classes)
```

Resultado:

```
> confusionMatrix(final_predict, dataset$classes)
Confusion Matrix and Statistics

                    Reference
Prediction           red soil cotton crop grey soil damp grey soil vegetation stubble very damp grey soil
  red soil               1533          0         0              0                   0                   0
  cotton crop               0        703         0              0                   0                   0
  grey soil                 0          0      1358              0                   0                   0
  damp grey soil            0          0         0            626                   0                   0
  vegetation stubble        0          0         0              0                 707                   0
  very damp grey soil       0          0         0              0                   0                1508

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9994, 1)
    No Information Rate : 0.2382
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: red soil Class: cotton crop Class: grey soil Class: damp grey soil Class: vegetation stubble
Sensitivity                   1.0000            1.0000           1.000              1.00000                    1.0000
Specificity                   1.0000            1.0000           1.000              1.00000                    1.0000
Pos Pred Value                1.0000            1.0000           1.000              1.00000                    1.0000
Neg Pred Value                1.0000            1.0000           1.000              1.00000                    1.0000
Prevalence                    0.2382            0.1092           0.211              0.09728                    0.1099
Detection Rate                0.2382            0.1092           0.211              0.09728                    0.1099
Detection Prevalence          0.2382            0.1092           0.211              0.09728                    0.1099
Balanced Accuracy             1.0000            1.0000           1.000              1.00000                    1.0000
                     Class: very damp grey soil
Sensitivity                              1.0000
Specificity                              1.0000
Pos Pred Value                           1.0000
Neg Pred Value                           1.0000
Prevalence                               0.2343
Detection Rate                           0.2343
Detection Prevalence                     0.2343
Balanced Accuracy                        1.0000
```

4. **Analise o resultado.**

A partir das predições feitas é possível afirmar que o modelo **RandomForest** teve a maior acurácia (0.9245). E, observando o modelo final gerado, a matriz de confusão alcançou 1 de acurácia, o máximo de acurácia que é possível atingir.

5. **Salve este modelo final**

Salvar modelo final:

```
saveRDS(final_model, "rf_satellite_final_model.rds")
```

Salvar o script:

```
setwd("C:/Projects/ia_ufpr/linguagem_r/trabalho_final/1_pesquisa_dados_satelite")
getwd()
save(final_model, file="satellite_commands.RData")
```

2 Estimativa de Volumes de Árvores

Modelos de aprendizado de máquina são bastante usados na área da engenharia florestal (mensuração florestal) para, por exemplo, estimar o volume de madeira de árvores sem ser necessário abatê-las.

O processo é feito pela coleta de dados (dados observados) através do abate de algumas árvores, onde sua altura, diâmetro na altura do peito (dap), etc, são medidos de forma exata. Com estes dados, treina-se um modelo de AM que pode estimar o volume de outras árvores da população.

Os modelos, chamados **alométricos**, são usados na área há muitos anos e são baseados em regressão (linear ou não) para encontrar uma equação que descreve os dados. Por exemplo, o modelo de Spurr é dado por:

$$\text{Volume} = b_0 + b_1 * dap^2 * H_t$$

Onde **dap** é o diâmetro na altura do peito (1,3 metros), $\mathbf{H_t}$ é a altura total. Tem-se vários modelos alométricos, cada um com uma determinada característica, parâmetros, etc. Um modelo de regressão envolve aplicar os dados observados e encontrar b_0 e b_1 no modelo apresentado, gerando assim uma equação que pode ser usada para prever o volume de outras árvores.

Dado o arquivo **Volumes.csv**, que contém os dados de observação, escolha um modelo de aprendizado de máquina com a melhor estimativa, a partir da estatística de correlação.

O arquivo `2_comandos.R` possui todos os comandos listados da atividade.

Adiciona o pacote *caret* pra fazer o particionamento dos dados:

```
install.packages("caret")
library("caret")
```

Tarefas:

1. **Carregar o arquivo Volumes.csv (http://www.razer.net.br/datasets/Volumes.csv):**

```
getwd()
dataset <-
read.csv("/Projects/ia_ufpr/linguagem_r/trabalho_final/2_estimativa_volumes_arvores/Volumes.csv", header = TRUE, sep = ';', dec = ',')
```

Visualizar os dados:

```
head(dataset)
```

Resultado:

```
    NR    DAP    HT       HP      VOL
1   1     34.0   27.00    1.80    0.8971441
2   2     41.5   27.95    2.75    1.6204441
3   3     29.6   26.35    1.15    0.8008181
4   4     34.3   27.15    1.95    1.0791682
5   5     34.5   26.20    1.00    0.9801112
6   6     29.9   27.10    1.90    0.9067022
```

2. **Eliminar a coluna NR, que só apresenta um número sequencial:**

```
dataset <- dataset[-1]
head(dataset)
```

Resultado:

```
    DAP     HT        HP      VOL
1   34.0    27.00     1.80    0.8971441
2   41.5    27.95     2.75    1.6204441
3   29.6    26.35     1.15    0.8008181
4   34.3    27.15     1.95    1.0791682
5   34.5    26.20     1.00    0.9801112
6   29.9    27.10     1.90    0.9067022
```

3. **Criar partição de dados: treinamento 80%, teste 20%:**

```
indexes <- createDataPartition(dataset$VOL, p = 0.80, list = FALSE)
```

4. **Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de SPURR:**

```
training <- dataset[indexes,]
test <- dataset[-indexes,]

set.seed(0)
rf <- train(VOL ~., data = training, method = 'rf', trControl =
trainControl('cv', number = 10), preProcess =     c('center', 'scale'))

svm <- train(VOL ~., data = training, method = 'svmRadial', trControl =
trainControl('cv', number = 10), preProcess = c('center', 'scale'))
```

```
nn <- train(VOL ~ ., data = training, method = 'neuralnet',
linear.output = TRUE, threshold = 0.1, trControl = trainControl('cv',
number = 10), preProcess = c('center', 'scale'))
```

5. **O modelo alométrico é dado por: Volume = b_0 + b_1 * dap^2 * H_t**

```
alom <- nls(VOL ~ b0 + b1 * DAP * DAP * HT, training, start = list(b0 =
0.5, b1 = 0.5))
```

6. **Efetue as predições nos dados de teste:**

```
predict.rf <- predict(rf, test)
predict.svm <- predict(svm, test)
predict.nn <- predict(nn, test)
predict.alom <- predict(alom, test)
```

7. **Crie funções e calcule as seguintes métricas entre a predição e os dados observados:**
   - Coeficiente de determinação: R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \widehat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

onde y_i é o valor observado, $\widehat{y_i}$ é o valor predito e $\bar{y}$ é a média dos valores y_i observados. Quanto mais perto de 1 melhor é o modelo;

```
r2 <- function(observations, predictions) {
  return (1 - (sum((test$VOL - predictions) ^ 2) / sum((test$VOL -
mean(test$VOL)) ^ 2)))
}
```

   - Erro padrão da estimativa: S_{yx}

$$S_{yx} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n-2}}$$

esta métrica indica erro, portanto quanto mais perto de 0 melhor é o modelo;

```
syx <- function(observations, predictions) {
  return (sqrt((sum((test$VOL - predictions) ^ 2) / (length(test$VOL) -
2))))
}
```

- $S_{yx}\%$

$$S_{yx}\% = \frac{S_{yx}}{\bar{y}} * 100$$

esta métrica indica porcentagem de erro, portanto quanto mais perto de 0 melhor é o modelo;

```
syx_percent <- function(observations, predictions) {
  return (syx(observations, predictions) / mean(test$VOL) * 100)
}
```

8. **Escolha o melhor modelo:**

Calcula métricas para o modelo baseado em Random Forest:

```
rf_r2 <- r2(test$VOL, predict.rf)
rf_syx <- syx(test$VOL, predict.rf)
rf_syx_percent <- syx_percent(test$VOL, predict.rf)
```

Calcula métricas para o modelo baseado em SVM:

```
svm_r2 <- r2(test$VOL, predict.svm)
svm_syx <- syx(test$VOL, predict.svm)
svm_syx_percent <- syx_percent(test$VOL, predict.svm)
```

Calcula métricas para o modelo baseado em Neural Network:

```
nn_r2 <- r2(test$VOL, predict.nn)
```

```
nn_syx <- syx(test$VOL, predict.nn)
nn_syx_percent <- syx_percent(test$VOL, predict.nn)
```

Calcula métricas para o modelo alométrico de Spurr:

```
alom_r2 <- r2(test_data$VOL, predict.alom)
alom_syx <- syx(test_data$VOL, predict.alom)
alom_syx_percent <- syx_percent(test_data$VOL, predict.alom)
```

Análise do modelo final:

```
final_model <- data.frame('RF' = c(rf_r2, rf_syx, rf_syx_percent),
           'SVM' = c(svm_r2, svm_syx, svm_syx_percent),
           'NN' = c(nn_r2, nn_syx, nn_syx_percent),
           'ALOM' = c(alom_r2, alom_syx, alom_syx_percent),
           row.names = c('$R^2$', 'Syx','Sxy%'))
final_model
```

Resultado:

```
         RF          SVM          NN          ALOM
 $R^2$   0.8180272   0.6470441   0.8400591   0.9319330
 Syx     0.2213733   0.3083061   0.2075400   0.1353912
 Sxy%    15.8802745  22.1164298  14.8879408  9.7123257
```

Salvar modelo final:

```
saveRDS(final_model, "volume_arvores_final_model.rds")
```

Salvar este script:

```
setwd("C:/Projects/ia_ufpr/linguagem_r/trabalho_final/2_estimativa_volumes_arvores")
getwd()
save(final_model, file="volumes_arvores_commands.RData")
```

O **modelo alométrico** é o que obteve os melhores resultados de acordo com os cálculos das funções: *coeficiente de determinação* e o *erro padrão da estimativa* (com apenas 9.7% de erro).