Inteligência Artificial
Terceiro Trabalho - Aprendizado de Máquina Supervisionado

Gabriel de Souza Alves - 726515
Isadora Eliziário Gallerani - 726542

1. Introdução

O presente trabalho consta da execução dos algoritmos de aprendizado de máquina supervisionado, sendo estes os de árvore de decisão e regressão. Foi disponibilizado e permitido o uso de funções disponíveis em pacotes de R.

O objetivo é executar o treinamento de classificadores, construindo dois tipos de árvores - uma de classificação e outra de regressão - para dois *datasets* distintos, utilizando a linguagem R.

Os datasets foram escolhidos com base no número de atributos (procuramos escolher conjuntos com poucos atributos, diminuindo a complexidade da aplicação do algoritmo), a qual atividade estes eram associados (classificação, regressão, etc) tendo em vista as informações do repositório UCI.

Para cada etapa, divide-se o *dataset* determinado de modo aleatório, em 75% de dados para o treinamento e os 25% restantes para testes. Em seguida, lê-se os arquivos de treinamento e teste usando a função *read.table()*. Para o conjunto de treinamentos, aplica-se *rpart()* e constrói a árvore gerada utilizando *rpart.plot()*. Tendo esta implementação em mãos, executa-se o modelo nos dados de testes, com *predict().*

Por fim, a partir dos dados obtidos, calcula-se a métrica de acurácia e precisão para a árvore de decisão para classificação, e o erro médio quadrático para a árvore de regressão.

2. Desenvolvimento

2.1. Árvore de Decisão para Classificação

Para esta etapa, foi selecionado o conjunto de dados "*Blood Transfusion Service Center Data Set*".

Este dataset de um banco de sangue é bastante conciso e simples, tendo 5 atributos (Meses desde a última doação, Número total de doações, Quantidade total de sangue doado, Meses desde a primeira doação) e 2 classes possíveis, indicando se o indivíduo fez uma doação ou não baseado nos dados citados.

Inicialmente, carrega-se o *dataset* em questão em um *data frame*, e escolhe valores aleatórios para este, levando em consideração a necessidade de uso de 75% de dados para treinamento e 25% para testes.

A função `createDataPartition()` do pacote `caret` já faz a separação entre classes igualmente, gerando uma amostra aleatória do conjunto com a mesma proporção entre classes (no caso, 50% de cada), sendo que 75% do conjunto inteiro seria utilizado para treinamento e 25% para testes.

```
dataFrame <- read.table('datasets/Blood-Transfusion-Dataset.csv', header =
TRUE, sep = ",")

sample <- createDataPartition(dataFrame$Donated, p = 0.75, list = FALSE)
```

Como o resultado guardado em *sample* é um vetor booleano, para escolher os candidatos que serão utilizados no conjunto de testes basta utilizar sua versão negada (considerando que o valor original foi utilizado para escolher o conjunto de treinamento).

```
trainSample <- dataFrame[sample,]
testSample <- dataFrame[-sample,]
```

Geramos então a árvore de decisão para o conjunto de dados de treinamento.

```
model <- rpart(formula = Donated ~ Recency + Frequency + Monetary + Time,
               data = trainSample, method = "class",
               control = rpart.control(minsplit = 1),
               parms = list(split = "Information"), model = TRUE)
```

Em seguida, realiza-se a etapa de predição no conjunto de testes, com o parâmetro *type* = *“class”* visto que queríamos o resultado de classificação.

```
predicted <- predict(model, testSample, type = "class")
```

Com o resultado da predição podemos montar a matriz de confusão do modelo, que nos dará a base para calcular as métricas dos resultados da predição contra os dados reais.

```
confMat <- table(predicted=predicted, truth=testSample$Donated)
                accuracy <- accuracy(confMat)
                precision <- precision(confMat))
```

Por fim, carrega-se os dados coletados em uma lista, a qual será retornada no final.

```
result <- list()
result$model <- model
result$accuracy <- accuracy
result$precision <- precision
return(result)
```

Ao final da execução do algoritmo, foi necessário trabalhar com as métricas deste caso de árvore. A matriz de confusão deste modelo estava na forma abaixo, um pouco diferente dos resultados comuns (onde os positivos ficam na primeira linha)

$$\begin{bmatrix} TN & FN \\ TP & FP \end{bmatrix}$$

Figura 1: Matriz de Confusão do Modelo

Para a precisão, considera-se como a soma dos positivos verdadeiros dividida pela soma de todos os valores positivos.

```
precision <- function(confusionMatr) {
  return(sum(confusionMatr[2,2])/sum(confusionMatr[2,]))
}
```

Já para a acurácia, considera-se a soma dos positivos verdadeiros e falsos, dividida pela soma de todos existentes.

```
accuracy <- function(confusionMatr) {
  return(sum(diag(confusionMatr))/sum(confusionMatr))
}
```

2.2. Árvore de Regressão

Para esta etapa, foi selecionado o conjunto de dados "*Servo Motor Data Set*".

Este *dataset* era igualmente simples, com 4 atributos sobre modelos de servomotores (Modelo do Motor, Modelo do parafuso usado no motor, Pgain e Vgain) e classifica o tempo de resposta (ou *rise time*) de cada indivíduo em um valor contínuo.

Como já fora feito no algoritmo anterior, inicialmente, carrega-se o *dataset* em questão em um *data frame*, e são escolhidos valores aleatórios para este, levando em consideração a necessidade de uso de 75% de dados para treinamento e 25% para testes. Como aqui os valores são contínuos não há possibilidade de estratificação do conjunto, então basta separar o conjunto em 75% e 25%.

Constrói-se a árvore de regressão para o conjunto de treinamento em questão.

```
model <- rpart(formula = Class ~ Motor + Screw + PGain + VGain,
                 data = trainSample, method = "anova", model = TRUE)
```

Realiza-se a predição no conjunto de dados e gera-se o modelo em questão, usando desta vez o parâmetro *type = vector* para indicar que queremos o resultado em um vetor de valores contínuos

```
predicted <- predict(model, testSample, type = "vector")
```

Em seguida, calcula-se o erro quadrático médio a partir do vetor de predição e do vetor de dados reais.

```
error <- mse(testSample$Class, predicted)
```

Por fim, carrega-se os dados coletados, os quais serão retornados em uma lista.

```
result <- list()
result$model <- model
result$mse <- error

return(result)
```

Ao final da execução do algoritmo, foi necessário trabalhar com o erro quadrático médio deste caso de árvore, o qual foi tratado como sendo a média da subtração entre dados reais e dado preditos, elevada ao quadrado.

```
mse <- function(real, predicted) {
  return(mean(real-predicted)^2)
}
```

3. Resultados

Este foi o resultado obtido para uma das instâncias executadas do algoritmo de classificação:

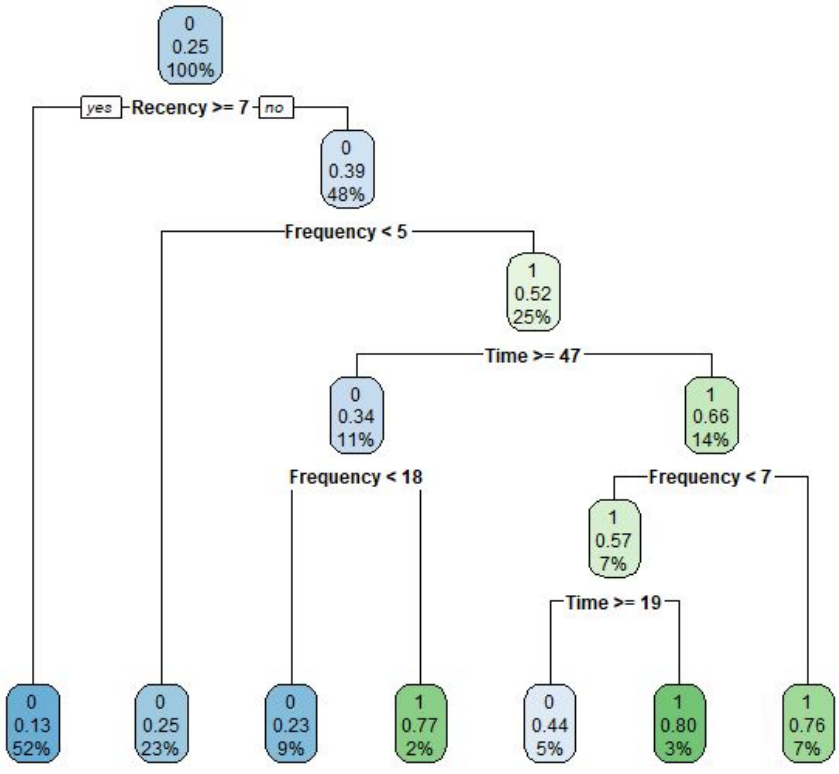

Figura 2: Resultado de uma instância do modelo de classificação

E a partir deste foram calculadas as seguintes métricas:

```
Acurácia:  0.8128342
Precisão:  0.5
```

No segundo processo, obtivemos o seguinte resultado em uma das instâncias:

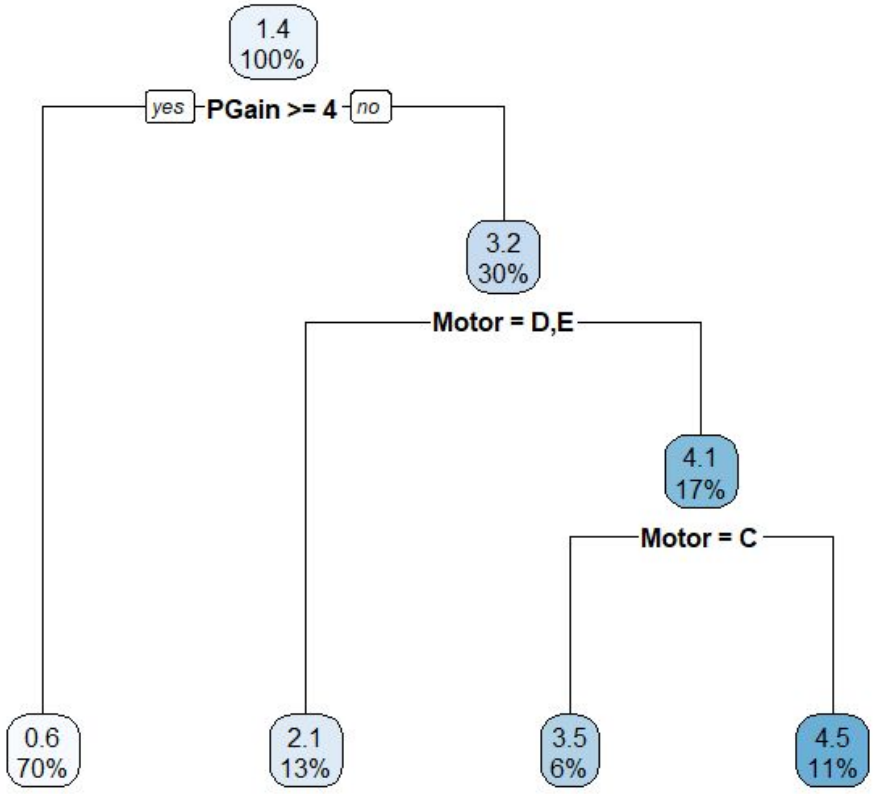

Figura 3: Resultado de uma instância do modelo de regressão

Na qual foi obtida a métrica de erro seguinte:

```
Erro Quadrado Médio:  0.01081946
```

4. Conclusão

Após este estudo foi possível ver que, mesmo com conjuntos de dados de poucos atributos aparentemente simples e bem comportados, ao usar *slices* aleatórios para treinamento e teste podem ser obtidos modelos bastante diferentes um do outro, assim como as métricas resultantes destes. Por isso, é

importante não utilizar como base o resultado de apenas uma instância do problema, mas sim tomar a média de resultados de várias instâncias (método bastante utilizado em vários métodos de validação) ou algum método mais complexo de validação de modelos, como a Validação Cruzada para que se possam obter resultados confiáveis sobre o classificador estudado.