Introdução à datascience com R

Cleuton Sampaio

Lição 8: Classificação

Classificação

Esta aula é sobre classificação. O que é classificação? Segundo a Wikipedia:

> *Em classificação, entradas são divididas em duas ou mais classes, e o aprendiz deve produzir um modelo que vincula entradas não vistas a uma ou mais dessas classes (classificação multi-etiquetada). Isso é tipicamente abordado de forma supervisionada. A filtragem de spam é um exemplo de classificação, em que as entradas são as mensagens de emails (ou outros) e as classes são "spam" ou "não spam".*

Os problemas de classificação são aqueles em que desejamos prever uma variável categórica. Podem ser de classificação binária, caso a categoria a ser prevista tenha apenas dois valores (sim e não, verdadeiro ou falso).

Podemos ter casos em que desejamos classificar dados em categorias multivaloradas, usando vários tipos de entradas, como: Imagens e Sons, por exemplo.

Neste capítulo, veremos uma técnica de supervised learning.

Existem vários algoritmos para classificação, entre eles:

- Regressão logística;
- Árvores de decisão;
- SVM – Support Vector Machine.

Neste material, vamos utilizar a Regressão logística.

Regressão logística

O modelo da regressão logística é dado pela fórmula:

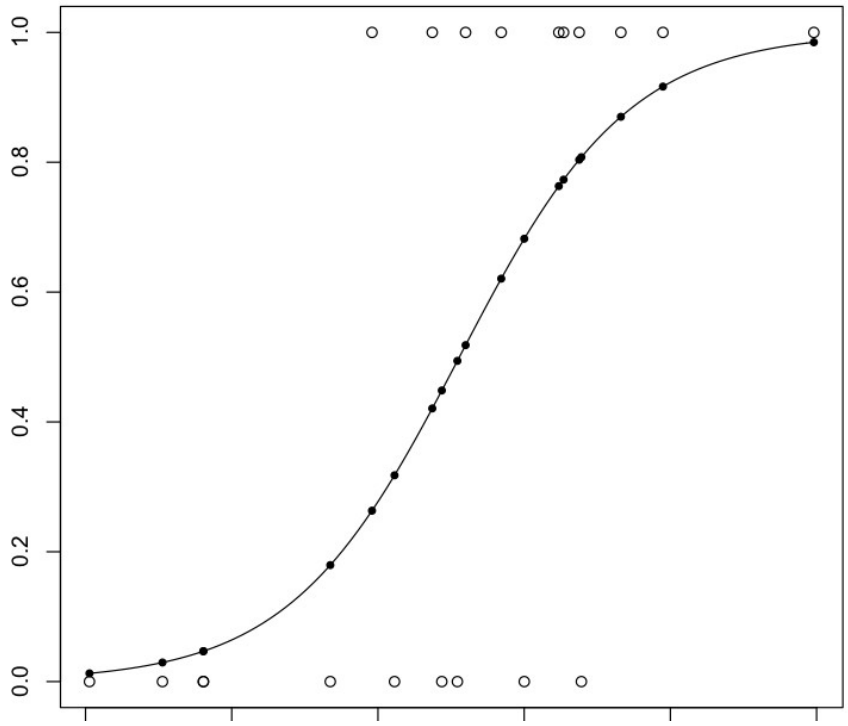
$$p_i = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+...+\beta_k x_k)}}$$

Onde

$$\beta_0+\beta_1 x_1+...+\beta_k x_k$$

É a combinação linear das variáveis independentes, cujos coeficientes são calculados pelo método da verossimilhança.

O gráfico da função logística é algo como um “s”:



O valor de resposta varia de zero a um, sendo a probabilidade dele ser mais próximo de zero ou um. Nós o arredondamos para ter um valor binomial.

Churn prediction

O Script desta aula está no repositório, com o nome “churn_prediction.R”.

Um problema popular de classificação é o “Churn prediction”, ou “Predição de evasão”. Pode ser aplicado a diversos modelos de negócio, como: Escolas, Cursos ou Serviços continuados. Todo aquele negócio em que o relacionamento com o cliente é longo e dependemos dele para faturar.

Há algum tempo, quando eu lecionava em uma instituição de ensino, passamos por uma situação em que o “churn” (a evasão) de alunos aumentou bastante. Fui encarregado de entender os motivos do abandono e criar medidas proativas para reter esses alunos.

Então, depois de um levantamento intenso, coletamos amostras e realizamos um estudo que nos permitiu criar um modelo preditivo, com o qual, orientamos campanhas de retenção de alunos em risco de evasão, com medidas como: ofertas de bolsas, aulas de reforço e trabalhos extraordinários, para recuperar as médias.

Eu tentei recuperar o máximo que pude dos dados e tive que filtrar várias coisas, para evitar identificação do curso. É claro que isso afetou um pouco os resultados, porém, mesmo assim, pode ser utilizado para explicar classificação binária.

O objetivo é obter um modelo que ao ser alimentado com dados de alunos, retorne um valor binário, indicando se sim (1) ele está no grupo de risco de evasão, ou não (0) ele não está no grupo de risco de evasão.

Antes de continuar, preciso falar um pouco mais sobre como selecionamos os alunos e suas características. Para começar, retiramos da amostra os alunos beneficiados com bolsas maiores que 50% de desconto, pois estes são controlados por sistemas diferentes e perdem a bolsa automaticamente.

Depois, especificamos o período de coleta, que deveria ser semanal, iniciando-se logo após as notas da primeira prova terem sido lançadas.

As características do dataset (“evasao.csv”) são:

- “periodo” : O período em que o aluno está no momento da coleta;
- “bolsa” : Percentual de bolsa de estudos com o qual o aluno foi beneficiado;
- “repetiu” : Quantas disciplinas o aluno falhou em aprovação;
- “ematraso” : Se o aluno está com mensalidades atrasadas;
- “disciplinas” : Quantas disciplinas o aluno está cursando no período da coleta;
- “faltas” : Quantas faltas o aluno teve no período, até o momento da coleta;
- “desempenho” : A mediana das notas das disciplinas que ele está cursando no período da coleta;
- “abandonou” : Se o aluno abandonou o curso ou não.

Coletamos os dados em dois períodos e depois os utilizamos para prever a evasão no período seguinte, e fomos acompanhando por algum tempo.

Para considerar se o aluno abandonou ou não o curso, quando ele trancou a matrícula, não reabriu a matrícula ou passou a faltar muitas aulas, sendo reprovado por isto.

Houve alguns problemas no levantamento que certamente afetaram o modelo, mas, como eu disse, nós o utilizamos com sucesso evitando cerca de 30% de evasão no período em que começamos a usá-lo para orientar campanhas proativas de recuperação de alunos.

Preparando os dados

Precisamos verificar como está o dataset, por exemplo, há dados faltando? Precisaremos igualar a escala dos preditores? Uma boa maneira de começar é essa:

```
df <- read.csv('evasao.csv')
print(head(df))
print(str(df))
print(summary(df))
```

Temos esse resultado da função “str()”:

```
'data.frame':     300 obs. of  8 variables:
 $ periodo    : int  2 2 4 4 1 5 9 2 9 5 ...
 $ bolsa      : num  0.25 0.15 0.1 0.2 0.2 0.2 0.1 0.15 0.15 0.15 ...
 $ repetiu    : int  8 3 0 8 3 2 6 3 7 3 ...
 $ ematraso   : int  1 1 1 1 1 1 1 0 1 0 ...
 $ disciplinas: int  4 3 1 1 1 2 1 2 5 1 ...
 $ faltas     : int  0 6 0 0 1 0 1 2 10 1 ...
 $ desempenho : num  0 5.33 8 4 8 ...
 $ abandonou  : int  1 0 0 1 0 1 0 1 0 0 ...
```

E temos esses resultados da função “summary()”:

```
    periodo          bolsa           repetiu         ematraso
 Min.   : 1.00   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
 1st Qu.: 3.00   1st Qu.:0.0500   1st Qu.:0.000   1st Qu.:0.0000
 Median : 5.00   Median :0.1000   Median :2.000   Median :0.0000
 Mean   : 5.46   Mean   :0.1233   Mean   :2.777   Mean   :0.4767
 3rd Qu.: 8.00   3rd Qu.:0.2000   3rd Qu.:5.000   3rd Qu.:1.0000
 Max.   :10.00   Max.   :0.2500   Max.   :8.000   Max.   :1.0000
  disciplinas        faltas         desempenho       abandonou
 Min.   :0.000   Min.   : 0.000   Min.   : 0.000   Min.   :0.00
 1st Qu.:1.000   1st Qu.: 0.000   1st Qu.: 0.400   1st Qu.:0.00
 Median :2.000   Median : 1.000   Median : 2.000   Median :0.00
 Mean   :2.293   Mean   : 2.213   Mean   : 2.623   Mean   :0.41
 3rd Qu.:4.000   3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.:1.00
 Max.   :5.000   Max.   :10.000   Max.   :10.000   Max.   :1.00
```

Como podemos verificar se há linhas com valores faltando? Uma maneira é usar a função “table()”:

```
table(is.na(df))
```

Se estiver faltando algum valor em alguma linha, esta função mostrará TRUE. No nosso caso, mostrou FALSE, então não precisamos nos preocupar com isso.

Quanto à escala, como vamos usar Regressão logística, isso não é importante. Porém, se você for utilizar outros algoritmos, como SVM, por exemplo, é melhor padronizar a escala dos valores dos preditores.

Separando dados de treino e de teste

Na vida real, nem sempre temos como coletar dados a todo momento. E, como vamos “treinar” o modelo, queremos saber o quão adequado ele está. Uma boa prática é separar nossa amostra em dois conjuntos de dados: Treino e Teste. Nós preparamos o modelo com os dados de Treino e o avaliamos com os dados de Teste. Assim, poderemos testá-lo com dados que ele nunca “viu”.

Em R é bem simples separar os dados:

```
n <- nrow(df)
limite <- sample(1:n, size = round(0.75*n), replace = FALSE)
train_df <- df[limite,]
test_df <- df[-limite,]
```

Estamos usando a função “sample()” para pegar 75% das linhas do dataframe original, separando-as para treino do modelo, e deixando 25% finais para teste. O resultado da função “sample()” é um vetor de índices. Atribuímos a “train_df” as linhas cujo índice está neste vetor, e a “test_df”, as linhas cujo índice não está no vetor.

Criando e treinando o modelo

Em R é muito simples:

```
modelo <- glm('abandonou ~ periodo + bolsa + repetiu + ematraso + disciplinas +
faltas + desempenho', data = train_df, family = 'binomial')
print(summary(modelo))
```

Parece muito com a função “lm()”, certo? A diferença é o parâmetro “family”, que estamos passando como “binomial”, indicando que desejamos um modelo logístico (logit). O “summary()” do modelo nos mostra coisas interessantes sobre ele:

```
Call:
glm(formula = "abandonou ~ periodo + bolsa + repetiu + ematraso + disciplinas +
faltas + desempenho",
    family = "binomial", data = train_df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9222  -0.8398  -0.4779   1.0002   1.8850

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.98098    0.55087  -1.781  0.07494 .
periodo     -0.05322    0.05225  -1.018  0.30845
bolsa       -0.41749    1.80151  -0.232  0.81674
repetiu      0.35847    0.06675   5.371 7.85e-08 ***
ematraso     0.42299    0.31203   1.356  0.17523
disciplinas  0.12463    0.10832   1.151  0.24991
faltas       0.01296    0.06561   0.198  0.84342
desempenho  -0.21289    0.06667  -3.193  0.00141 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 306.45  on 224  degrees of freedom
Residual deviance: 254.13  on 217  degrees of freedom
AIC: 270.13
```

Repare as legendas de significância: Somente dois preditores efetivamente contribuem para o resultado: “repetiu” e “desempenho”, o resto pode ser apenas coincidência.

Podemos rodar um teste ANOVA com a distribuição Qui-quadrado para confirmar nossas suspeitas:

```
print(anova(modelo, test = "Chisq"))
```

```
           Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                         224     306.45
periodo     1    1.587       223     304.86 0.2078042
bolsa       1    0.314       222     304.55 0.5750856
repetiu     1   35.359       221     269.19 2.743e-09 ***
ematraso    1    1.470       220     267.72 0.2253472
disciplinas 1    2.036       219     265.68 0.1536110
faltas      1    0.013       218     265.67 0.9102985
desempenho  1   11.537       217     254.13 0.0006822 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Somente as linhas que contém os asterisco são relevantes. Podemos construir um novo modelo retirando as outras variáveis:

```
train_r <- subset(train_df,select = c('repetiu','desempenho','abandonou'))

modelo2 <- glm('abandonou ~ repetiu + desempenho', data = train_r, family =
'binomial')
print(summary(modelo2))
print(anova(modelo2, test = "Chisq"))
```

Sempre é bom retirarmos aquelas variáveis que pouco contribuem para o resultado geral, afinal de contas essa “contribuição” pode ser apenas coincidência, e nós ficamos com um modelo menor, contendo somente: “desempenho” e “repetiu”.

Testando o modelo

Vamos fazer predições e testar o modelo:

```
test_r <- subset(test_df,select = c('repetiu','desempenho','abandonou'))
resultados <- predict(modelo2,newdata = test_r, type = 'response')
resultados_ar <- ifelse(resultados > 0.5, 1, 0)
erroMedio <- mean(resultados_ar != test_r$abandonou)
print(paste('Precisão modelo reduzido:',1 – erroMedio))
```

Primeiro, retiramos do dataframe de teste as variáveis que não contribuem, deixando apenas “repetiu”, “desempenho” e a variável dependente “abandonou”.

Depois, usamos a função “predict()”, indicando que desejamos as probabilidades de cada valor de resposta.

O resultado do “predict()” é algo assim:

```
0.87181691 0.20206125 0.34027381 0.59299934 0.22195667 0.72600538 0.47992454
```

Mas não é isso que desejamos, certo? Queremos saber se ele: 1 – abandonará ou 0 – não abandonará. Então, arredondamos as previsões:

```
resultados_ar <- ifelse(resultados > 0.5, 1, 0)
```

Sempre que o valor retornado for maior que 0,5, retorne “1”, senão, retorne zero.

Agora, precisamos saber o quanto erramos. Podemos comparar com os valores reais da coluna “abandonou”, dos dados de teste:

```
erroMedio <- mean(resultados_ar != test_r$abandonou)
```

E calculamos o quando acertamos (1 – percentual de erros):

```
Precisão modelo reduzido: 0.613333333333333
```

Acertamos 61% dos valores. Nada mal. Seu resultado pode ser ligeiramente diferente, pois estamos selecionando os dados aleatoriamente (“sample()”).

Agora, vamos comparar com o modelo original, sem retirar as variáveis:

```
resultados2 <- predict(modelo,newdata = test_df, type = 'response')
resultados2_ar <- ifelse(resultados2 > 0.5, 1, 0)
erroMedio2 <- mean(resultados2_ar != test_df$abandonou)
print(paste('Precisão modelo original:',1 – erroMedio2))
```

```
Precisão modelo original: 0.666666666666667
```

Bem, temos um resultado ligeiramente melhor. Isto é explicado pelo fato de que as variáveis que retiramos estão, de fato contribuindo, mesmo que pouco. Se você vai manter o modelo original ou não, a decisão é sua. Como a diferença é muito pequena, vale a pena ficarmos com um modelo menor, com apenas 3 dimensões (“abandonou”, “repetiu” e “desempenho”).

Conclusão

Conseguimos criar um modelo que acertou mais de 60% dos casos de abandono. Então, podemos implementar uma política de dar maior atenção aos alunos que repetiram muitas cadeiras e estão com desempenho ruim.