Indicações de Soluções das Atividades de Aprofundamento

Trilha 06

Atividade de Aprofundamento

Problema 1

Vamos fazer um exercício de Regressão Logística, trabalhando com a base de dados do Titanic.

Utilizaremos os arquivos disponibilizados no Moodle que foram preparados para esta atividade. Embora os nomes sejam os mesmos, estes arquivos do Moodle são diferentes (em conteúdo) daqueles encontrados no site Kaggle.

A base de dados (treinamento) é uma coleção de dados sobre alguns dos passageiros (916 para ser preciso) e o objetivo é prever a sobrevivência (1 se o passageiro sobreviveu ou 0 caso contrário), baseado em algumas características tais como classe de serviço, sexo, idade, etc. A base de dados de teste tem a mesma estrutura da base de treinamento, mas a variável `survived` não contém nenhum valor.

Depois de construir seu modelo, você vai aplicá-lo ao este conjunto de dados de teste e prever o valor da variável `survived`.

Como se pode ver, utilizaremos tanto variáveis categÃ³ricas como contínuas.

Quando trabalhamos com uma base de dados real, precisamos levar em conta o fato de podermos ter dados faltantes ou corrompidos, e, portanto, precisamos preparar a base de dados para nossa análise.

```
library('dplyr') # manipulacao de dados

train.raw <- read.csv('D:\\Aulas\\Mackenzie\\Análise Estatística\\Datasets\\titanic_train.csv', stringsAsFactors = F,na.strings = c(""))
test  <- read.csv('D:\\Aulas\\Mackenzie\\Análise Estatística\\Datasets\\titanic_test.csv', stringsAsFactors = F, na.strings=c(""))

full  <- bind_rows(train.raw, test) # juntamos tudo para algumas inspecoes.

# primeira inspecao
str(full,strict.width="wrap")
'data.frame':   1309 obs. of  12 variables:
$ PassengerId: int 1 2 3 4 5 6 7 8 9 10 ...
$ Survived : int 0 1 1 1 0 0 0 0 1 1 ...
$ Pclass : int 3 1 3 1 3 3 1 3 3 2 ...
$ Name : chr "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley
   (Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs.
   Jacques Heath (Lily May Peel)" ...
```

```
$ Sex : chr "male" "female" "female" "female" ...
$ Age : num 22 38 26 35 35 NA 54 2 27 14 ...
$ SibSp : int 1 1 0 1 0 0 0 3 0 1 ...
$ Parch : int 0 0 0 0 0 0 0 1 2 0 ...
$ Ticket : chr "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
$ Fare : num 7.25 71.28 7.92 53.1 8.05 ...
$ Cabin : chr NA "C85" NA "C123" ...
$ Embarked : chr "S" "C" "S" "S" ...
```

As tarefas a serem realizadas são:

a) Fazer a preparação dos dados para serem utilizados na análise, considerando que serão utilizadas apenas as variáveis `survived`, `pclass`, `sex`, `age`, `sibsp`, `parch`, `fare`, `embarked`:

```
# número de dados faltantes em cada variável
sapply(train.raw,function(x) sum(is.na(x)))
PassengerId    Survived      Pclass        Name         Sex         Age
          0           0           0           0           0         177
      SibSp       Parch      Ticket        Fare       Cabin    Embarked
          0           0           0           0         687           2

# quantos valores diferentes em cada variável
sapply(train.raw, function(x) length(unique(x)))
PassengerId    Survived      Pclass        Name         Sex         Age
        891           2           3         891           2          89
      SibSp       Parch      Ticket        Fare       Cabin    Embarked
          7           7         681         248         148           4
```

Como podemos ver, a variável `Cabin` tem muitos valores faltantes – não a utilizaremos em nossa análise.

Podemos utilizar a função `subset()` para selecionar as colunas relevantes; também podemos utilizar a função `select()` do pacote `dplyr` para a mesma finalidade.

i) Do conjunto de dados original, você deve selecionar um subconjunto apenas com as variáveis indicadas acima e a variável `PassengerId`.

```
data <- train.raw %>% select(c(1,2,3,5,6,7,8,10,12))
```

ii) Você deve atribuir um valor para os *NAs* na variável age. Utilize algum critério razoável, por exemplo, o valor médio.

```
data <- data %>% mutate(Age = ifelse(is.na(Age), mean(Age,na.rm=T),Age))
```

iii) Você deve remover as linhas onde ainda estiverem faltando dados, depois de atribuir o valor para os *NAs* de age. Poucas linhas estarão ainda com dados faltantes.

Nova verificação:

```
# número de dados faltantes em cada variável
sapply(data,function(x) sum(is.na(x)))
PassengerId    Survived      Pclass         Sex         Age       SibSp
          0           0           0           0           0           0
      Parch        Fare    Embarked
          0           0           2
```

A função `complete.cases()` pode ser utilizada para inspecionarmos as linhas faltantes. Ela retorna um vetor de booleanos (TRUE ou FALSE) que utilizamos para selecionar as linhas desejadas (completas). Como queremos apenas as linhas faltantes, utilizamos a negação **!** para pegar apenas aquelas linhas que **não** estão completas:

```
data[!complete.cases(data),]
    PassengerId Survived Pclass    Sex Age SibSp Parch Fare Embarked
62           62        1      1 female  38     0     0   80     <NA>
830         830        1      1 female  62     0     0   80     <NA>
```

Observamos que as linhas 62 e 830 são as que contém NAs na variável *Embarked*.

Como temos apenas dois valores faltantes, então vamos desprezar estas duas linhas (também poderíamos substituir os NAs pela moda, por exemplo e manter todos as observaçÃµes).

```
data <- data[complete.cases(data),]

# para remover os nomes das linhas
rownames(data) <- NULL

head(data)
  PassengerId Survived Pclass    Sex      Age SibSp Parch    Fare Embarked
1           1        0      3   male 22.00000     1     0  7.2500        S
2           2        1      1 female 38.00000     1     0 71.2833        C
3           3        1      3 female 26.00000     0     0  7.9250        S
4           4        1      1 female 35.00000     1     0 53.1000        S
5           5        0      3   male 35.00000     0     0  8.0500        S
6           6        0      3   male 29.69912     0     0  8.4583        Q
```

b) Você deve criar um modelo onde survived será uma função das demais variáveis.

```
model <- glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare + Embarked,
family=binomial(link='logit'),data=data)
summary(model)

Call:
glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Parch +
    Fare + Embarked, family = binomial(link = "logit"), data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6446  -0.5907  -0.4230   0.6220   2.4431

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.285188   0.564778   9.358  < 2e-16 ***
Pclass      -1.100058   0.143529  -7.664 1.80e-14 ***
Sexmale     -2.718695   0.200783 -13.540  < 2e-16 ***
Age         -0.039901   0.007854  -5.080 3.77e-07 ***
SibSp       -0.325777   0.109384  -2.978   0.0029 **
Parch       -0.092602   0.118708  -0.780   0.4353
Fare         0.001918   0.002376   0.807   0.4194
EmbarkedQ   -0.034076   0.381936  -0.089   0.9289
EmbarkedS   -0.418817   0.236794  -1.769   0.0769 .
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1182.82  on 888  degrees of freedom
Residual deviance:  784.19  on 880  degrees of freedom
AIC: 802.19

Number of Fisher Scoring iterations: 5
```

c) Faça as análises do modelo, verificando a significÃ¢ncia estatística das variáveis (e seus parÃ¢metros ajustados), gráficos diagnÃ³sticos, etc.

Olhando o resultado do sumário do nosso modelo, verificamos que algumas variáveis não apresentam significÃ¢ncia estatística: `Parch`, `Fare`, `Embarked`.

Para cada parÃ¢metro que ajustamos no modelo, o **R** nos dá o intervalo de confiança de tais valores ajustados através da função `confint`. Quando rodamos esta função no modelo ajustado, nossa análise deve ser verificar se algum parÃ¢metro tem no seu intervalo de confiança o *zero*, ou seja, se o intervalo **contém** o zero. Quando isso acontece temos uma indicação de que este parÃ¢metro não tem significÃ¢ncia estatística, já que ele poderia ter o valor *zero*.

```
confint(model)
                   2.5 %       97.5 %
(Intercept)  4.198048925  6.415646275
Pclass      -1.384464864 -0.820594213
Sexmale     -3.121536866 -2.333494538
Age         -0.055590504 -0.024758983
SibSp       -0.550931743 -0.121861284
Parch       -0.331589735  0.137207868
Fare        -0.002492127  0.007071418
EmbarkedQ   -0.786934758  0.712394517
EmbarkedS   -0.882645200  0.046861201
```

Do resultado acima, confirmamos que as variáveis `Parch`, `Fare` e `Embarked` contém o *zero* nos seus respectivos intervalos de confiança.

Uma regressão logística utiliza a função `generalized linear models`, justamente porque nossa variável alvo `Survived` é do tipo binária: 1 ou 0. Então não podemos utilizar os gráficos diagnÃ³sticos anteriores; ficamos apenas com a avaliação da significÃ¢ncia estatística dos parÃ¢metros.

Uma avaliação que podemos fazer é sobre "Superdispersão"

```
 deviance(model)/df.residual(model)
[1] 0.8911235
```

Como o valor obtido é bem prÃ³ximo de 1, podemos assumir que não temos "superdispersão" neste modelo.

d) Atualize o modelo como consequência da análise realizada no item anterior.

Vamos começar removendo a variável `Embarked`, que não apresentou significÃ¢ncia estatística – aliás, era esperado este resultado, não?!?

```
model <- glm(Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare, family = b
inomial(link ='logit'), data = data)
```

```
summary(model)

Call:
glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Parch +
    Fare, family = binomial(link = "logit"), data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7116  -0.6031  -0.4285   0.6202   2.4173

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.955763   0.532777   9.302  < 2e-16 ***
Pclass      -1.082309   0.139051  -7.784 7.05e-15 ***
Sexmale     -2.755710   0.199192 -13.834  < 2e-16 ***
Age         -0.039907   0.007813  -5.108 3.26e-07 ***
SibSp       -0.349905   0.109437  -3.197  0.00139 **
Parch       -0.110417   0.117378  -0.941  0.34686
Fare         0.002822   0.002355   1.198  0.23081
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1182.82  on 888  degrees of freedom
Residual deviance:  788.22  on 882  degrees of freedom
AIC: 802.22

Number of Fisher Scoring iterations: 5

 deviance(model)/df.residual(model)
[1] 0.8936791
```

Valor prÃ³ximo de 1, indicando ausência de “superdispersão”.

Nossa variável a ser removida é Fare

```
model <- glm(Survived ~ Pclass + Sex + Age + SibSp + Parch, family = binomial
(link ='logit'), data = data)
summary(model)

Call:
glm(formula = Survived ~ Pclass + Sex + Age + SibSp + Parch,
    family = binomial(link = "logit"), data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6529  -0.6145  -0.4234   0.6168   2.4309

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
```

```
(Intercept)  5.233838   0.483620  10.822  < 2e-16 ***
Pclass      -1.169917   0.119711  -9.773  < 2e-16 ***
Sexmale     -2.760660   0.198903 -13.879  < 2e-16 ***
Age         -0.040314   0.007794  -5.172 2.31e-07 ***
SibSp       -0.333668   0.108445  -3.077  0.00209 **
Parch       -0.080343   0.114667  -0.701  0.48351
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1182.8  on 888  degrees of freedom
Residual deviance:  789.8  on 883  degrees of freedom
AIC: 801.8

Number of Fisher Scoring iterations: 5

 deviance(model)/df.residual(model)
[1] 0.8944541
```

Valor prÃ³ximo de 1, indicando ausência de “superdispersão”.

Como visto, a variável Parch continua sem significÃ¢ncia estatística, e portanto, deve ser removida.

```
model <- glm(Survived ~ Pclass + Sex + Age + SibSp, family = binomial(link ='
logit'), data = data)
summary(model)

Call:
glm(formula = Survived ~ Pclass + Sex + Age + SibSp, family = binomial(link =
"logit"),
    data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6857  -0.6055  -0.4215   0.6125   2.4528

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.185020   0.478347  10.839  < 2e-16 ***
Pclass      -1.169400   0.119747  -9.766  < 2e-16 ***
Sexmale     -2.732586   0.194302 -14.064  < 2e-16 ***
Age         -0.040013   0.007772  -5.148 2.63e-07 ***
SibSp       -0.356735   0.103918  -3.433 0.000597 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1182.8  on 888  degrees of freedom
Residual deviance:  790.3  on 884  degrees of freedom
```

```
AIC: 800.3

Number of Fisher Scoring iterations: 5
```

Obtemos agora um modelo em que todos os parÃ¢metros ajustados tem significÃ¢ncia estatística.

```
 deviance(model)/df.residual(model)
[1] 0.8940046
```

Valor prÃ³ximo de 1, indicando ausência de “superdispersão”.

e) Faça as previsÃµes da variável `survived` na base de dados de teste utilizando o modelo refinado, e prepare um arquivo CSV para submissão que contenha apenas duas colunas: `passId`, `survived`.

```
# número de dados faltantes em cada variável
sapply(test,function(x) sum(is.na(x)))
PassengerId      Pclass        Name         Sex         Age       SibSp
          0           0           0           0          86           0
      Parch      Ticket        Fare       Cabin    Embarked
          0           0           1         327           0

# atribuindo o valor médio para os NA's da variável Age
test <- test %>% mutate(Age = ifelse(is.na(Age), mean(Age,na.rm=T),Age))

sapply(test,function(x) sum(is.na(x)))
PassengerId      Pclass        Name         Sex         Age       SibSp
          0           0           0           0           0           0
      Parch      Ticket        Fare       Cabin    Embarked
          0           0           1         327           0

results.Survived <- predict(model,newdata=subset(test,select=c(2,4,5,6,7,9,11
)),type='response')
results.Survived <- ifelse(results.Survived  > 0.5,1,0)
table(results.Survived)
results.Survived
  0   1
257 161
```

f) Submeta seu arquivo e também o script **R** com todas as análises realizadas, até a criação do arquivo CSV.

```
submissao <- data.frame(PassengerId = test[,"PassengerId"], Survived = result
s.Survived)
write.csv(submissao, "D:\\Aulas\\Mackenzie\\Análise Estatística\\Datasets\\ti
tanic-submissao.csv",row.names = FALSE, quote = FALSE)
```