Testagem de COVID-19 em SP

Comparação de Modelos Preditivos

Bruno de Castro Paul Schultze, Lucas Bortolucci, Lucas Monteiro Bianchi, Luiz Afonso Glatzl Junior, T

12/11/2020

Sumário

Introdução

Durante a pandemia de Covid-19, ficou-se evidente a importância da testagem em massa, entretanto, muitos testes necessitam de equipamentos sofisticados e de execução especializada. Diante disso, os testes rápidos ganharam certa notoriedade e vem sendo largamente utilizados. Esses testes são geralmente sorológicos, capazes de identificar os anticorpos produzidos quando em contato com o antígeno. A metodologia mais utilizada por esse tipo de teste é o imunocromatografia, no qual, se consiste em produzir uma coloração através da reação entre o antígeno e o anticorpo, sendo capazes de detectar se o organismo esta porduzindo anticorpos (IgG e IgM). Embora as vantagens desse tipo de teste são claras, os testes rápidos

possuem baixa sensibilidade. Podemos definir sensibilidade como a probabilidade do teste resultar em positivo quando o indíviduo de fato tem a doença. Em outras palavras, um teste com baixa sensibilidade pode resultar em um elevado número de falsos negativos, gerando estatísticas que não expressam a realidade e consequentemente dificultam o processo de tomada de decisões.

Recentemente, a vigilância da COVID-19, a infecção humana causada pelo novo Coronavírus, que vem causando uma pandemia, foi incorporada na rede de vigilância da Influenza e outros vírus respiratórios. Os dados são provenientes do site openDataSUS. Esta página tem como finalidade disponibilizar o legado dos banco de dados (BD) epidemiológicos de SRAG, da rede de vigilância da Influenza e outros vírus respiratórios, desde o início da sua implantação (2009) até os dias atuais (2020), com a incorporação da vigilância da COVID-19.

Objetivos

- Identificar a capacidade de prevermos o resultado positivo de uma testagem para Covid-19
- Implementar e comparar diferentes modelos de classificação vistos na disciplina

Preparação dos dados

Dados

A população definida para esse estudo são pessoas residentes do estado de São Paulo cujo foram diagnosticadas com SRAG-Covid 19. Segundo o Ministério da Saúde, temos as seguintes definições:

Sindrome gripal (SG): Indivíduo com quadro respiratório agudo, caracterizado por pelo menos dois (2) dos seguintes sinais e sintomas: febre (mesmo que referida), calafrios, dor de garganta, dor de cabeça, tosse, coriza, distúrbios olfativos ou distúrbios gustativos. Observações:

- Em crianças: além dos itens anteriores considera-se também obstrução nasal, na ausência de outro diagnóstico específico.
- Em idosos: deve-se considerar também critérios específicos de agravamento como sincope, confusão mental, sonolência excessiva, irritabilidade e inapetência.
- Na suspeita de COVID-19, a febre pode estar ausente e sintomas gastrointestinais (diarreia) podem estar presentes.

Síndrome Respiratória Aguda Grave (SRAG): Indivíduo com SG que apresente: dispneia/desconforto respiratório OU pressão persistente no tórax OU saturação de O_2 menor que 95% em ar ambiente OU coloração azulada dos lábios ou rosto.

Além disso, um caso de covid-19 pode ser confirmado por:

- Por critério clínico: Caso de SG ou SRAG com confirmação clínica associado a anosmia (disfunção olfativa) OU ageusia (disfunção gustatória) aguda sem outra causa pregressa.
- Por critério clínico-epidemiológico: Caso de SG ou SRAG com histórico de contato próximo ou domiciliar, nos 14 dias anteriores ao aparecimento dos sinais e sintomas com caso confirmado para COVID-19.

- Por critério clínico-imagem: Caso de SG ou SRAG ou óbito por SRAG que não foi possível confirmar por critério laboratorial E que apresente pelo menos uma (1) das seguintes alterações tomográficas: opacidade em vidro fosco (periférico ou multifocal) e sinal de halo reverso.

Observações: Em crianças: além dos itens anteriores, observar os batimentos de asa de nariz, cianose, tiragem intercostal, desidratação e inapetência; Para efeito de notificação no Sivep-Gripe, devem ser considerados os casos de SRAG hospitalizados ou os óbitos por SRAG independente de hospitalização.

Carregando pacotes

Carregando pacotes do R.

Importando o banco de dados

No código abaixo estamos importando os dados e aplicando os filtros necessárias paras as realizarmos as analises. Os dados utilizados podem ser baixados em OpenDataSus.

```
# Esse chunk é apenas para criar o banco de dados
# como demora um pouco, o resultado dessa saida foi salvo em dados.Rdata

# files <- dir() %>%
#   data.frame() %>%
#   filter(str_detect(.,"dados-sp-")) %>%
#   unlist() %>%
#   as.vector()
#
# dados <- NULL
# for(i in length(files)){
#   temp <- fread(files[i], header= TRUE, stringsAsFactors = T) %>%
#     filter(estadoIBGE == "35") %>%
#     filter(estado =="SÃO PAULO") %>%
#     filter(sexo %in% c("Masculino", "Feminino")) %>%
#     filter(resultadoTeste %in% c("Negativo", "Positivo")) %>%
#     filter(profissionalSaude %in% c("Não", "Sim")) %>%
#     mutate(
#       dataNotificacao = as.Date(substr(dataNotificacao,1,10), fmt = "%Y-%m-%d")
#     ) %>%
#     filter(dataNotificacao > lubridate::dmy("01-06-2020"))
#   dados <- rbind(dados,temp)
# }
# save(dados, file= "dados.Rdata")
```

```
load("dados.Rdata")
#write.csv(dados, file="dados.csv")

# Criando os sintomeas
dados <- dados %>%
  mutate(
    resultadoTeste = factor(resultadoTeste, levels = c("Positivo", "Negativo")),
    profissionalSaude = factor(profissionalSaude, levels = c("Não", "Sim")),
```

```
    Assintomatico = as.factor(ifelse(grepl("Assintomático",sintomas),1,0)),
    Coriza = as.factor(ifelse(grepl("Coriza",sintomas),1,0)),
    Dispneia = as.factor(ifelse(grepl("Dispneia",sintomas),1,0)),
    DistGustativos = as.factor(ifelse(grepl("Distúrbios Gustativos",sintomas),1,0)),
    DistOlfativos = as.factor(ifelse(grepl("Distúrbios Olfativos",sintomas),1,0)),
    DorDeCabeca = as.factor(ifelse(grepl("Dor de Cabeça",sintomas),1,0)),
    DorDeGarganta = as.factor(ifelse(grepl("Dor de Garganta",sintomas),1,0)),
    Febre = as.factor(ifelse(grepl("Febre",sintomas),1,0)),
    Tosse = as.factor(ifelse(grepl("Tosse",sintomas),1,0)),
    Outros = as.factor(ifelse(grepl("Outros",sintomas),1,0)),
    verificacao =
      as.numeric(Coriza) + as.numeric(Dispneia) + as.numeric(DistGustativos) +
      as.numeric(DistOlfativos) + as.numeric(DorDeCabeca) + as.numeric(DorDeGarganta) + as.numeric(Febre
    verificacao2 = case_when(
      Assintomatico == 1 & verificacao >= 1 ~ 1,
      TRUE ~ 0
    ),
    tipoTeste =  case_when(
      grepl("RÁPIDO",tipoTeste) == TRUE ~ "Teste rápido",
      grepl("RT-PCR",tipoTeste) == TRUE ~ "RT-PCR",
      TRUE ~ "Outro"
    ),
    idade = as.integer(as.character(idade)),
    faixaetaria = cut(idade,
                      breaks = c(seq(0,60,10),Inf), right = F,include.lowest = T,
                      levels = c("0 a 9", "10 a 19", "20 a 29", "30 a 39", "40 a 49", "50 a 59", ">= 60
  ) %>%
  filter(verificacao2 !=1) %>%
  filter(tipoTeste != "Outro") %>%
  filter(idade <= 100) %>%
  dplyr::select(-c(verificacao,verificacao2))

dados$resultadoTeste = relevel(dados$resultadoTeste, ref = 'Negativo')
# Selecionando apenas as informações que serão utilizadas
dados.modelo <- dados %>%
  filter(tipoTeste == "RT-PCR") %>%
  dplyr::select(resultadoTeste, sexo, faixaetaria, idade,
                Assintomatico,Coriza, Dispneia , DistGustativos,
                DistOlfativos, DorDeCabeca, DorDeGarganta,
                Febre, Tosse, Outros, profissionalSaude, dataNotificacao)

# Separar os dados de RT-PCR temporalmente, e os de teste serao os de teste rápido
dados.modelo <- dados.modelo[order(as.Date(dados.modelo$dataNotificacao, format="%Y-%m-%d")),] %>% dply
index_max <- (3*nrow(dados.modelo))%/%4 + 1
dados.treino <- dados.modelo[1:index_max]
dados.valid <- dados.modelo[(index_max+1):nrow(dados.modelo)]
dados.test <- dados %>%
  filter(tipoTeste != "RT-PCR") %>%
  dplyr::select(resultadoTeste, sexo, faixaetaria, idade,
                Assintomatico, Coriza, Dispneia , DistGustativos,
                DistOlfativos, DorDeCabeca, DorDeGarganta,
                Febre, Tosse, Outros, profissionalSaude)
```

Tabela 1: Estatística descritiva para cada características e sintomas dos indivíduos que compõem o conjunto de treinamento.

	level	Overall	Negativo	Positivo
n		212665	136921	75744
sexo (%)	Feminino	121371 (57.07)	80253 (58.61)	41118 (54.29)
	Masculino	91294 (42.93)	56668 (41.39)	34626 (45.71)
Assintomatico (%)	0	212665 (100.00)	136921 (100.00)	75744 (100.00)
	1	0 (0.00)	0 (0.00)	0 (0.00)
Dispneia (%)	0	172018 (80.89)	109029 (79.63)	62989 (83.16)
	1	40647 (19.11)	27892 (20.37)	12755 (16.84)
DistGustativos (%)	0	192381 (90.46)	128287 (93.69)	64094 (84.62)
	1	20284 (9.54)	8634 (6.31)	11650 (15.38)
DistOlfativos (%)	0	195248 (91.81)	130376 (95.22)	64872 (85.65)
	1	17417 (8.19)	6545 (4.78)	10872 (14.35)
DorDeCabeca (%)	0	148493 (69.82)	94391 (68.94)	54102 (71.43)
	1	64172 (30.18)	42530 (31.06)	21642 (28.57)
DorDeGarganta (%)	0	129895 (61.08)	80103 (58.50)	49792 (65.74)
	1	82770 (38.92)	56818 (41.50)	25952 (34.26)
Febre (%)	0	136274 (64.08)	92669 (67.68)	43605 (57.57)
	1	76391 (35.92)	44252 (32.32)	32139 (42.43)
Tosse (%)	0	94403 (44.39)	62770 (45.84)	31633 (41.76)
	1	118262 (55.61)	74151 (54.16)	44111 (58.24)
Outros (%)	0	98989 (46.55)	64560 (47.15)	34429 (45.45)
	1	113676 (53.45)	72361 (52.85)	41315 (54.55)
profissionalSaude (%)	Não	198576 (93.38)	127548 (93.15)	71028 (93.77)
	Sim	14089 (6.62)	9373 (6.85)	4716 (6.23)

```
# Apgando objetos desnecessarios
rm(dados, dados.modelo, index_max)
```

```
fatores <- colnames(dados.treino)[c(1:2,7:16)]

variaveis <- colnames(dados.treino)[c(2,5,7:16)]

## Criando a tabela 1 - Treino
tableOne.treino <- CreateTableOne(vars = variaveis, strata = "resultadoTeste", data = dados.treino, fac
tabela1.treino <- print(tableOne.treino, showAllLevels = T, test = F, catDigits = 2,printToggle = F)
```

A tabela abaixo apresenta as características e sintomas dos individuos que integram o conjunto de dados de treinamento.

```
tabela1.treino %>%
  knitr::kable(caption = "Estatística descritiva para cada características e sintomas dos indivíduos qu
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

```
## Criando a tabela 1 - Validacao
tableOne.valid <- CreateTableOne(vars = variaveis, strata = "resultadoTeste", data = dados.valid, facto
tabela1.valid <- print(tableOne.valid, showAllLevels = T, test = F, catDigits = 2,printToggle = F)
```

Tabela 2: Estatística descritiva para cada características e sintomas dos indivíduos que compõem o conjunto de validação.

	level	Overall	Negativo	Positivo
n		70887	48528	22359
sexo (%)	Feminino	40002 (56.43)	27941 (57.58)	12061 (53.94)
	Masculino	30885 (43.57)	20587 (42.42)	10298 (46.06)
Assintomatico (%)	0	70887 (100.00)	48528 (100.00)	22359 (100.00)
	1	0 (0.00)	0 (0.00)	0 (0.00)
Dispneia (%)	0	58648 (82.73)	39636 (81.68)	19012 (85.03)
	1	12239 (17.27)	8892 (18.32)	3347 (14.97)
DistGustativos (%)	0	61706 (87.05)	44544 (91.79)	17162 (76.76)
	1	9181 (12.95)	3984 (8.21)	5197 (23.24)
DistOlfativos (%)	0	62904 (88.74)	45448 (93.65)	17456 (78.07)
	1	7983 (11.26)	3080 (6.35)	4903 (21.93)
DorDeCabeca (%)	0	41447 (58.47)	28400 (58.52)	13047 (58.35)
	1	29440 (41.53)	20128 (41.48)	9312 (41.65)
DorDeGarganta (%)	0	42623 (60.13)	27948 (57.59)	14675 (65.63)
	1	28264 (39.87)	20580 (42.41)	7684 (34.37)
Febre (%)	0	46434 (65.50)	33080 (68.17)	13354 (59.73)
	1	24453 (34.50)	15448 (31.83)	9005 (40.27)
Tosse (%)	0	33373 (47.08)	23259 (47.93)	10114 (45.23)
	1	37514 (52.92)	25269 (52.07)	12245 (54.77)
Outros (%)	0	38435 (54.22)	26318 (54.23)	12117 (54.19)
	1	32452 (45.78)	22210 (45.77)	10242 (45.81)
profissionalSaude (%)	Não	67489 (95.21)	46133 (95.06)	21356 (95.51)
	Sim	3398 (4.79)	2395 (4.94)	1003 (4.49)

A tabela abaixo apresenta as características e sintomas dos individuos que integram o conjunto de dados de validação. É possivel notar a semelhança com os valores obtidos na tabela anterior para o conjunto de treinamento, ressaltando que a aleatorização foi devidamente realizada.

```
tabela1.valid %>%
  knitr::kable(caption = "Estatística descritiva para cada características e sintomas dos indivíduos qu
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

A tabela abaixo apresenta os "resultados" obtidos através do uso do teste rápido. É importante ressaltar que mesmo tendo os "resultados" para o banco de teste, este trabalho desconsidera essa informação, visto que este tipo de teste é menos confiavel que o teste RT-PCR. De toda forma, essa informação é relevante para comparar com o cenário onde todos os individuos do banco teste teriam realizado o RT-PCR ao invés do teste rápido, permitindo desta forma, obter uma estimativa do quanto o teste rápido, possívelmente, submestiou os casos positivos de covid-19.

```
prop.table(table(dados.test$resultadoTeste)) %>%
  knitr::kable(caption = "Proporção dos resultados obtidos utilizando o teste rápido para Covid-19 (ban
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

Tabela 3: Proporção dos resultados obtidos utilizando o teste rápido para Covid-19 (banco teste).

Var1	Freq
Negativo	0.7425741
Positivo	0.2574259

```
# Removendo os assintomaticos do conjunto treinamento, validação e teste
# pois essa variavel é colinear quando considerados todos os demais sintomas
dados.treino <- dados.treino %>% dplyr::select(-Assintomatico)
dados.valid  <-dados.valid %>% dplyr::select(-Assintomatico)
dados.test <- dados.test %>% dplyr::select(-Assintomatico)
```

Metodologias

Nesta seção é apresentado o conjunto de metodologias empregadas para classificar, segundo os sintomas, se a pessoa é positivo ou negativo para covid-19.

Foram utilizados os seguintes métodos: * Regressão Logistica * Random Forest * Suporte Vector Machine * Redes Neurais Artificiais

Métricas de qualidade dos modelos

Foram utilizadas as seguintes métricas

- Sensibilidade: Probabilidade do resultado ser positivo quando de fato o individuo tem a doença.
- Especificidade: Probabilidade do resultado ser negativo quando de fato o individuo não tem a doença.
- Acurácia: probabilidade do teste fornecer resultados corretos, ou seja, ser positivo nos doentes e negativo nos não doentes. Expresso de outra forma é a probabilidade dos verdadeiros positivos e verdadeiros negativos como uma proporção de todos osresultados

Para esse estudo, ao invés de considerarmos o "individuo ter ou não ter a doença", teremos "o teste RT-PCR positivo ou negativo", ficando a reinterpretação da sensibilidade e especificidade dessas métricas conforme abaixo:

- Sensibilidade: Probabilidade do modelo resultar em positivo quando o teste RT-PCR foi positivo.
- Especificidade: Probabilidade do modelo resultar em negativo quando o teste RT-PCR foi negativo.

Regressão Logística

Aqui, assumimos que o resultado do teste dadas as variáveis explicativas, $Y|x_i$, assume uma distribuição *Bernoulli* com probabilidade π_i de ser positivo.

$$Y|x_i \sim Bernoulli(\pi_i)$$

Além disso, a função de ligação é:

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + +\beta_{13} x_{13}$$

Primeiro vamos considerar o modelo completo contendo como variáveis explicativas:

- Profissional da saúde
- Faixa Etária
- Idade
- Sexo
- Sintomas (uma variável indicadora para cada)

```
fit.rl = glm(resultadoTeste ~ ., family = binomial,data = dados.treino)
summary(fit.rl)
```

```
##
## Call:
## glm(formula = resultadoTeste ~ ., family = binomial, data = dados.treino)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9885  -0.9231  -0.7585   1.2661   2.2885
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.776547   0.031964 -55.580  < 2e-16 ***
## sexoMasculino         0.150314   0.009568  15.709  < 2e-16 ***
## faixaetaria[10,20)    0.542912   0.038044  14.271  < 2e-16 ***
## faixaetaria[20,30)    0.543762   0.039993  13.596  < 2e-16 ***
## faixaetaria[30,40)    0.769901   0.048037  16.027  < 2e-16 ***
## faixaetaria[40,50)    0.805843   0.057445  14.028  < 2e-16 ***
## faixaetaria[50,60)    0.898586   0.068126  13.190  < 2e-16 ***
## faixaetaria[60,Inf]   0.766303   0.084743   9.043  < 2e-16 ***
## idade                 0.004391   0.001193   3.679 0.000234 ***
## Coriza1              -0.256682   0.012323 -20.829  < 2e-16 ***
## Dispneia1            -0.318016   0.012334 -25.783  < 2e-16 ***
## DistGustativos1       0.492395   0.021047  23.395  < 2e-16 ***
## DistOlfativos1        0.980779   0.022587  43.423  < 2e-16 ***
## DorDeCabeca1         -0.153650   0.010749 -14.295  < 2e-16 ***
## DorDeGarganta1       -0.301171   0.009978 -30.185  < 2e-16 ***
## Febre1                0.493495   0.009810  50.308  < 2e-16 ***
## Tosse1                0.233627   0.009806  23.824  < 2e-16 ***
## Outros1               0.118902   0.009843  12.079  < 2e-16 ***
## profissionalSaudeSim -0.054746   0.019306  -2.836 0.004573 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 276966  on 212664  degrees of freedom
## Residual deviance: 262592  on 212646  degrees of freedom
```

```
## AIC: 262630
##
## Number of Fisher Scoring iterations: 4
```

Seleção do Modelo

Vamos usar stepAIC para tentar escolher um subconjunto das variáveis explicativas.

```
fit.rl = stepAIC(fit.rl, direction = 'both', trace = 0)
summary(fit.rl)
```

```
##
## Call:
## glm(formula = resultadoTeste ~ sexo + faixaetaria + idade + Coriza +
##     Dispneia + DistGustativos + DistOlfativos + DorDeCabeca +
##     DorDeGarganta + Febre + Tosse + Outros + profissionalSaude,
##     family = binomial, data = dados.treino)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9885  -0.9231  -0.7585   1.2661   2.2885
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.776547   0.031964 -55.580  < 2e-16 ***
## sexoMasculino         0.150314   0.009568  15.709  < 2e-16 ***
## faixaetaria[10,20)    0.542912   0.038044  14.271  < 2e-16 ***
## faixaetaria[20,30)    0.543762   0.039993  13.596  < 2e-16 ***
## faixaetaria[30,40)    0.769901   0.048037  16.027  < 2e-16 ***
## faixaetaria[40,50)    0.805843   0.057445  14.028  < 2e-16 ***
## faixaetaria[50,60)    0.898586   0.068126  13.190  < 2e-16 ***
## faixaetaria[60,Inf]   0.766303   0.084743   9.043  < 2e-16 ***
## idade                 0.004391   0.001193   3.679 0.000234 ***
## Coriza1              -0.256682   0.012323 -20.829  < 2e-16 ***
## Dispneia1            -0.318016   0.012334 -25.783  < 2e-16 ***
## DistGustativos1       0.492395   0.021047  23.395  < 2e-16 ***
## DistOlfativos1        0.980779   0.022587  43.423  < 2e-16 ***
## DorDeCabeca1         -0.153650   0.010749 -14.295  < 2e-16 ***
## DorDeGarganta1       -0.301171   0.009978 -30.185  < 2e-16 ***
## Febre1                0.493495   0.009810  50.308  < 2e-16 ***
## Tosse1                0.233627   0.009806  23.824  < 2e-16 ***
## Outros1               0.118902   0.009843  12.079  < 2e-16 ***
## profissionalSaudeSim -0.054746   0.019306  -2.836 0.004573 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 276966  on 212664  degrees of freedom
## Residual deviance: 262592  on 212646  degrees of freedom
## AIC: 262630
##
```

```
## Number of Fisher Scoring iterations: 4
```

Todas as variáveis foram mantidas pelo algoritmo. Vamos então checar por multicolinearidade.

```
car::vif(fit.rl)
```

```
##                        GVIF Df GVIF^(1/(2*Df))
## sexo                1.026810  1        1.013316
## faixaetaria        18.714348  6        1.276482
## idade              17.966002  1        4.238632
## Coriza              1.074528  1        1.036594
## Dispneia            1.023820  1        1.011840
## DistGustativos      1.814785  1        1.347140
## DistOlfativos       1.817173  1        1.348026
## DorDeCabeca         1.081349  1        1.039879
## DorDeGarganta       1.053246  1        1.026278
## Febre               1.036171  1        1.017925
## Tosse               1.072933  1        1.035825
## Outros              1.093049  1        1.045490
## profissionalSaude   1.032012  1        1.015880
```

Notemos que há evidências de multicolinearidade usando os critérios usuais (VIF maiores que 5), então retiraremos *idade* do modelo.

```
fit.rl = glm(resultadoTeste ~ ., family = binomial,data = subset(dados.treino, select = -idade))
summary(fit.rl)
```

```
##
## Call:
## glm(formula = resultadoTeste ~ ., family = binomial, data = subset(dados.treino,
##     select = -idade))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9604  -0.9254  -0.7597   1.2651   2.2842
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.758550   0.031587 -55.673  < 2e-16 ***
## sexoMasculino         0.149936   0.009568  15.671  < 2e-16 ***
## faixaetaria[10,20)    0.596746   0.035124  16.990  < 2e-16 ***
## faixaetaria[20,30)    0.635299   0.031322  20.283  < 2e-16 ***
## faixaetaria[30,40)    0.904819   0.031044  29.146  < 2e-16 ***
## faixaetaria[40,50)    0.982872   0.031400  31.302  < 2e-16 ***
## faixaetaria[50,60)    1.119449   0.032234  34.728  < 2e-16 ***
## faixaetaria[60,Inf]   1.054347   0.032388  32.553  < 2e-16 ***
## Coriza1              -0.257096   0.012322 -20.864  < 2e-16 ***
## Dispneia1            -0.316857   0.012329 -25.700  < 2e-16 ***
## DistGustativos1       0.492322   0.021047  23.392  < 2e-16 ***
## DistOlfativos1        0.980232   0.022587  43.399  < 2e-16 ***
```

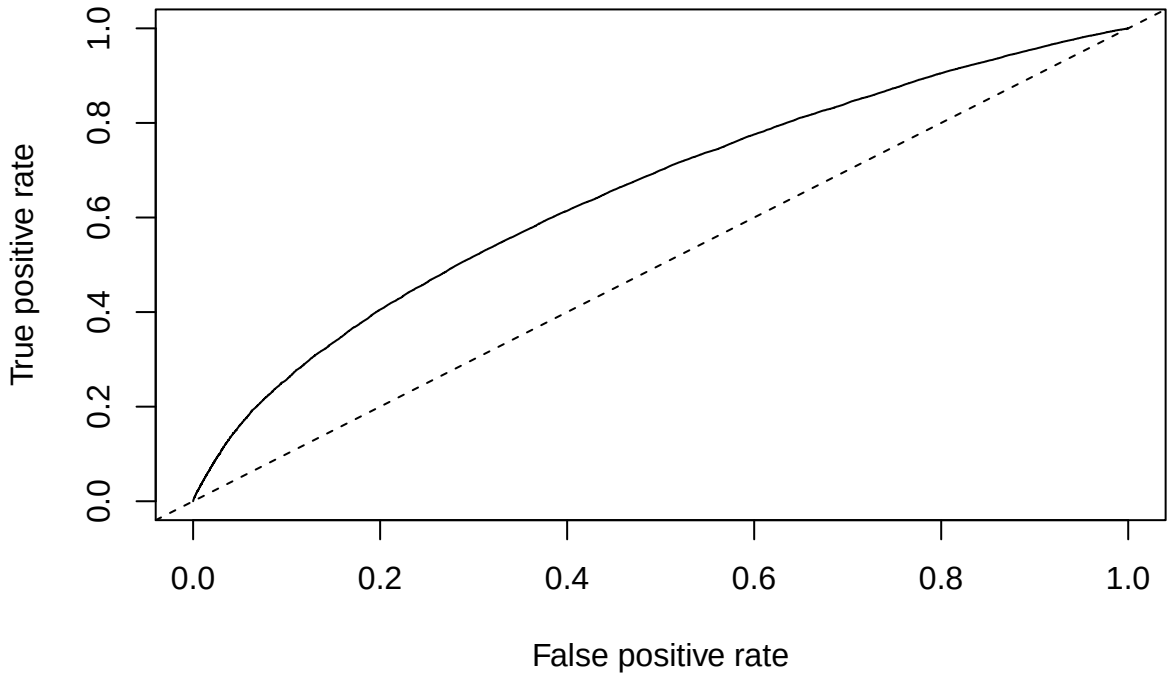
```
## DorDeCabeca1         -0.154592   0.010745 -14.387  < 2e-16 ***
## DorDeGarganta1       -0.302313   0.009972 -30.316  < 2e-16 ***
## Febre1                0.492841   0.009807  50.252  < 2e-16 ***
## Tosse1                0.233660   0.009806  23.828  < 2e-16 ***
## Outros1               0.118561   0.009842  12.046  < 2e-16 ***
## profissionalSaudeSim -0.055013   0.019307  -2.849  0.00438 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 276966  on 212664  degrees of freedom
## Residual deviance: 262605  on 212647  degrees of freedom
## AIC: 262641
##
## Number of Fisher Scoring iterations: 4
```

Com o modelo definido, partiremos então para a definição do ponto de corte. Para isso levaremos em conta principalmente Sensibilidade, mas também Especificidade e Acurácia Total.
Para definir o ponto de corte utilizaremos os dados de treino mesmo.

```
pred.rl.treino = predict(fit.rl, type = 'response')
```

Abaixo podemos ver a curva ROC:

```
pr.train = ROCR::prediction(pred.rl.treino, dados.treino$resultadoTeste)
pr.train.perf = ROCR::performance(pr.train, measure = 'tpr', x.measure = 'fpr')
plot(pr.train.perf)
abline(a=0.0, b= 1.0, lty = 2)
```



Vamos checar também a área embaixo da curva (AUC) que criamos anteriormente.

```
ROCR::performance(pr.train, measure = 'auc')@y.values[[1]]
```

```
## [1] 0.6505379
```

Agora vamos escolher o threshold.
Para isso primeiro vamos definir o "custo" de um erro de forma heterogênea: um Falso Negativo será maior que o peso de um Falso Positivo. Isso se dá pois é pior dizer para um paciente infectado que ele não está com COVID (pois isso fará com que ele não obedeça o isolamento social e a quarentena de forma tão incisiva) do que dizer para uma pessoa não-infectada que ela está com COVID (pois isso somente fará com que ela realize exames adicionais que mostrarão que a mesma não está infectada). Não alteraremos a prevalência dado que não temos o conhecimento técnico epidemiológico para tal decisão.
Para a escolha do threshold usaremos o critério de Youden com uma modificação proposta por Perkins e Schisterman, dado por:

$$max(sensitivities + r \times specificities)$$

Onde:

$$r = \frac{1 - prevalence}{cost * prevalence}$$

Sendo *cost* o custo relativo de um Falso Negativo comparado com o de um Falso Positivo e prevalence número de Positivos dividido pelo total.

```r
prop.table(table(dados.treino$resultadoTeste))
```

```
##
##  Negativo  Positivo
## 0.6438342 0.3561658
```

Comparação variando Custo

Para comparar e escolher o custo, testaremos duas possibilidades ($custo = 2$ e $custo = 3$), e escolheremos o custo cujo threshold maximiza a sensibilidade nos dados de treino.

Primeiro, testaremos com $custo = 2$.

```r
custo = 2
prevalencia = 0.3561658
rl.ROC <- roc(dados.treino$resultadoTeste, pred.rl.treino,plot=FALSE)
thre = as.numeric(coords(rl.ROC, "best", ret = "threshold", best.weights = c(cost = custo, prevalence =
thre
```

```
## [1] 0.3462531
```

Agora, com o *threshold* definido, veremos então a matriz de confusão com relação aos dados de treino.

```r
rl.treino = as.factor(ifelse(pred.rl.treino > thre,'Positivo','Negativo'))
confusionMatrix(rl.treino, reference = dados.treino$resultadoTeste, positive = 'Positivo')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negativo Positivo
##   Negativo    83151    29650
##   Positivo    53770    46094
##
##                Accuracy : 0.6077
##                  95% CI : (0.6057, 0.6098)
##     No Information Rate : 0.6438
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2015
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.6085
##             Specificity : 0.6073
##          Pos Pred Value : 0.4616
##          Neg Pred Value : 0.7371
##              Prevalence : 0.3562
##          Detection Rate : 0.2167
```

```
##    Detection Prevalence : 0.4696
##       Balanced Accuracy : 0.6079
##
##        'Positive' Class : Positivo
##
```

O modelo conseguiu classificar corretamente 61% (sensibilidade) das pessoas que tiveram resultado positivo no exame, e classificou corretamente 61% das pessoas que tiveram resultado negativo.

Agora, com custo = 3:

```
custo = 3
prevalencia = 0.3561658
rl.ROC <- roc(dados.treino$resultadoTeste, pred.rl.treino,plot=FALSE)
thre = as.numeric(coords(rl.ROC, "best", ret = "threshold", best.weights = c(cost = custo, prevalence =
thre
```

```
## [1] 0.2507847
```

Notemos que, como esperado, o threshold diminuiu. Veremos então a matriz de confusão com relação aos dados de treino.

```
rl.treino = as.factor(ifelse(pred.rl.treino > thre,'Positivo','Negativo'))
confusionMatrix(rl.treino, reference = dados.treino$resultadoTeste, positive = 'Positivo')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negativo Positivo
##   Negativo    30634     8232
##   Positivo   106287    67512
##
##                Accuracy : 0.4615
##                  95% CI : (0.4594, 0.4636)
##     No Information Rate : 0.6438
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0892
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8913
##             Specificity : 0.2237
##          Pos Pred Value : 0.3884
##          Neg Pred Value : 0.7882
##              Prevalence : 0.3562
##          Detection Rate : 0.3175
##    Detection Prevalence : 0.8172
##       Balanced Accuracy : 0.5575
##
##        'Positive' Class : Positivo
##
```

O modelo conseguiu classificar corretamente 89% (sensibilidade) das pessoas que tiveram resultado positivo no exame, e classificou corretamente 22% das pessoas que tiveram resultado negativo.

Escolheremos então *custo* = 3 pois, apesar de diminuir a especificidade, maximizou a capacidade do algoritmo de detectar positivos verdadeiros.

Avaliação do Modelo Final

Vamos checar as métricas agora olhando para os dados de validação (que são como nossos dados de teste).

```
pred.rl.val = predict(fit.rl, newdata = dados.valid, type = 'response')
rl.val <- as.factor(ifelse(pred.rl.val > thre,"Positivo","Negativo"))
confusionMatrix(rl.val, reference = dados.valid$resultadoTeste, positive = 'Positivo')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negativo Positivo
##   Negativo    13238     2993
##   Positivo    35290    19366
##
##                Accuracy : 0.4599
##                  95% CI : (0.4563, 0.4636)
##     No Information Rate : 0.6846
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8661
##             Specificity : 0.2728
##          Pos Pred Value : 0.3543
##          Neg Pred Value : 0.8156
##              Prevalence : 0.3154
##          Detection Rate : 0.2732
##    Detection Prevalence : 0.7710
##       Balanced Accuracy : 0.5695
##
##        'Positive' Class : Positivo
##
```

A sensibilidade novamente é de 87% enquanto a especificidade é de 27%, uma performance parecida com a dos dados de treino. Por fim, veremos o desempenho do modelo nos dados de treino, que usam testes menos poderosos e, portanto, podem apresentar erros.

```
pred.rl.test = predict(fit.rl, newdata = dados.test %>% dplyr::select(-resultadoTeste), type = 'respons
rl.test = as.factor(ifelse(pred.rl.test > thre,'Positivo','Negativo'))
confusionMatrix(rl.test, reference = dados.test$resultadoTeste, positive = 'Positivo')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Negativo Positivo
##   Negativo    24114     5429
##   Positivo   105209    39403
##
##                Accuracy : 0.3647
##                  95% CI : (0.3625, 0.367)
##     No Information Rate : 0.7426
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0378
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8789
##             Specificity : 0.1865
##          Pos Pred Value : 0.2725
##          Neg Pred Value : 0.8162
##              Prevalence : 0.2574
##          Detection Rate : 0.2263
##    Detection Prevalence : 0.8304
##       Balanced Accuracy : 0.5327
##
##        'Positive' Class : Positivo
##
```

Notemos, então, que a sensibilidade nos dados de teste foi de 88% e a especificidade de 19%. Apesar das métricas de teste terem sido parecidas, convem lembrar que não podemos usar as mesmas para avaliação do modelo pois os testes laboratoriais utilizados nas amostras não são tão confiáveis.

A proporção prevista para a base de testes foi:

```
prop.table(table(rl.test))
```

```
## rl.test
##  Negativo  Positivo
## 0.1696362 0.8303638
```

Por fim, criaremos um vetor que resume as métricas utilizadas nos três conjuntos de dados.

```
Acuracia = c(46.15, 45.99, 36.47)
Sensibilidade = c(89.13, 86.61, 87.89)
Especificidade = c(22.37, 27.28, 18.65)
med_rl = cbind(Acuracia, Sensibilidade, Especificidade)
rownames(med_rl) = c('Treino', 'Validação', 'Teste')
med_rl %>%
  knitr::kable(caption = "Métricas de qualidade do modelo logistico.") %>%
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

Tabela 4: Métricas de qualidade do modelo logistico.

	Acuracia	Sensibilidade	Especificidade
Treino	46.15	89.13	22.37
Validação	45.99	86.61	27.28
Teste	36.47	87.89	18.65

E o tempo de execução do *fit* é:

```
tic('RL fitting')
fit.rl2 = glm(resultadoTeste ~ ., family = binomial,data = dados.treino %>% dplyr::select(-idade))
toc()
```

```
## RL fitting: 3.64 sec elapsed
```

Random forest

Também conhecida em português por florestas aleatórias, este algoritmo estabelece regras para tomada de decisão por meio de uma estrutura similar a um fluxograma, onde condições são verificadas, e se atendida, o fluxo segue por um ramo, caso contrário, por outro, sempre levando ao próximo nó, até a finalização da árvore. Além disso, este método requer que seja escolhida variáveis para compor cada nó e isso é feito aleatório entre as variáveis disponiveis para então realizar os cálculos com base nas amostras selecionadas, definindo assim qual dessas variáveis será utilizada no primeiro nó. Para escolha da variável do próximo nó, novamente serão escolhidas outras variáveis, desconsiderando as já selecionadas anteriormente, e o processo de escolha se repetirá. Assim, a árvore será construída até o último nó.

```
#h2o.removeAll()
localH2O <- h2o.init(nthreads = -1)
```

```
##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     C:\Users\bruno\AppData\Local\Temp\Rtmpac8dto\file38447eed5330/h2o_bruno_started_from_r.out
##     C:\Users\bruno\AppData\Local\Temp\Rtmpac8dto\file38446f6c1dfc/h2o_bruno_started_from_r.err
##
##
## Starting H2O JVM and connecting: . Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         6 seconds 799 milliseconds
##     H2O cluster timezone:       America/Sao_Paulo
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    1 month and 24 days
##     H2O cluster name:           H2O_started_from_R_bruno_muq595
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.75 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
```

```
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)
```

```
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         7 seconds 162 milliseconds
##     H2O cluster timezone:       America/Sao_Paulo
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    1 month and 24 days
##     H2O cluster name:           H2O_started_from_R_bruno_muq595
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.75 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  8
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 4.0.3 (2020-10-10)
```

```
# Criando os conjuntos no padrao H2o
train.h2o <- as.h2o(dados.treino)
```

```
##   |                                                                      |
```

```
valid.h2o <- as.h2o(dados.valid)
```

```
##   |                                                                      |
```

```
test.h2o <- as.h2o(dados.test)
```

```
##   |                                                                      |
```

```
ini <- Sys.time()
rforest.model.h2o <- h2o.randomForest(y=1, x=c(2,4,7:13),
                                      training_frame = train.h2o,
                                      validation_frame = valid.h2o,
                                      ntrees = 1000, mtries = 3, max_depth = 4, seed = 1122)
```

Tabela 5: Métricas de qualidade do modelo Random Forest

	Acuracia	Sensibilidade	Especificidade
Treino	56.19	70.01	48.55
Validação	56.98	70.33	50.83
Teste	50.75	67.34	44.99

```
##   |                                                                      |
```

```
end <- Sys.time()

print(paste0("Time spent was: ",round(end-ini,2)))
```

```
## [1] "Time spent was: 1.17"
```

A tabela abaixo apresenta as métricas obtidas com o modelo Random Forest.

```
Acuracia = c(56.19, 56.98, 50.75)
Sensibilidade = c(70.01, 70.33, 67.34)
Especificidade = c(48.55, 50.83, 44.99)
med_rl = cbind(Acuracia, Sensibilidade, Especificidade)
rownames(med_rl) = c('Treino', 'Validação', 'Teste')
med_rl %>%
  knitr::kable(caption = "Métricas de qualidade do modelo Random Forest") %>%
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

Support Vector Machines

O modelo baseado em Support Vector Machines foi implementado na linguagem Python utilizando a biblioteca Scikit Learn, e na linguagem R utilizando a biblioteca E1071. No entanto, avaliando o desempenho computacional das bibliotecas, foi selecionada a linguagem Python para implementação e avaliando deste modelo.

Na utilização de um Perceptron, nem sempre obtemos a melhor fronteira de decisão entre os dados. Com o SVM, buscamos obter o hiperplano separador ótimo, ou seja, uma boa relação entre margem de valor elevado e poucos erros marginais.

Para um SVM Linear, podemos ter dois tipos de margens no modelo, rígida ou suave. Na margem rígida, o hiperplano separador é ajustado avaliando apenas os vetores suporte, não permitindo erros marginais. Já com margem suave, são permitidos um determinado número de amostras entre os vetores suporte, e que pode ser ajustado utilizando o parâmetro C, inversamente proporcional a margem.

No entanto, em grande parte das aplicações os dados não são linearmente separáveis. Para tais situações, podemos utilizar o Kernel Trick para realizar mudanças de espaço, obtendo assim uma fronteira de decisão linear que proporcione a obtenção de um hiperplano separador ótimo.

A implementação do modelo utilizada, bem como a etapa de validação e teste estão disponíveis no Github do grupo, o script denominado SVM.ipynb

Os resultado obtidos foram:

- Treino
  - AUC: 0.61
  - Acurácia: 0.59
  - Sensibilidade: 0.50
  - Especificidade: 0.69
- Validação
  - Acurácia: 0.65
  - Sensibilidade: 0.51
  - Especificidade: 0.71

Redes Neurais Artificiais

O modelo de redes neurais artificiais foi implementado em python com a biblioteca Keras (Tensorflow), com o acompanhamento do treino feito pelo Comet.ml. Em sua arquitetura, tratando-se de uma Rede Neural totalmente conectada, possui uma hidden layer (com 8 unidades), 137 parâmetros e função de ativação sigmoid:

$$\sigma(x) = \frac{1}{1 + e^x}$$

Outras caracteristicas da rede e de configuração de otimização são:

- Função de custo
  - Binnary Cross-Entropy
  - $H_P(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$
- Otimizador
  - Adam (Padrão)
- Batch size
  - 1024 indivíduos
- Epochs
  - 100
- Método de escolha do ponto de corte
  - Youden's J statistic

A implementação do método foi feita através do Python. Os arquivos relacionados estão disponíveis no Github (cujo link está no fim desse relatório). Para mais detalhes, veja os seguintes arquivo e link:

- RNA.ipynb (O arquivo Script_trabalho.Rmd deve ser rodado antes para o funcionamento ideal deste notebook)
- https://www.comet.ml/martinoni/mae5904-rna/a0c7e454164d4b11a6cd9a4882bba9f1 (Abas contando informações imporantes: Charts; Metrics; Graphics; Confusion Matrices)

Tabela 6: Métricas de qualidade nos dados de treino

Modelo	Acuracia	Sensibilidade	Especificidade
Reg. Logística	46.15	89.13	22.37
Random Forest	56.19	70.01	48.55
SVM	59.00	50.00	69.00
RNA	52.00	81.00	40.00

Com essas características os resultados obtidos foram:

- Treino
  - AUC: 0.67
  - Acurácia: 0.52
  - Sensibilidade: 0.81
  - Especificidade: 0.40
- Validação
  - Acurácia: 0.55
  - Sensibilidade: 0.83
  - Especificidade: 0.36

Comparações entre modelos

Para realizar a comparação entre os modelos foram utilizadas 3 principais métricas: acurária, sensibilidade e especificidade.

Nos dados de treino, temos:

```
treino_df <- data.frame(Modelo = c("Reg. Logística","Random Forest","SVM","RNA"),
                        Acuracia = c(46.15,56.19,59.00,52.00),
                        Sensibilidade = c(89.13,70.01,50.00,81.00),
                        Especificidade = c(22.37,48.55,69.00,40.00))

treino_df %>%
  knitr::kable(caption = "Métricas de qualidade nos dados de treino") %>%
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

Nos dados de validação, temos:

```
validacao_df <- data.frame(Modelo = c("Reg. Logística","Random Forest","SVM","RNA"),
                        Acuracia = c(45.99,56.98,65.00,55.00),
                        Sensibilidade = c(86.61,70.33,51.00,83.00),
                        Especificidade = c(27.28,50.83,71.00,36.00))

validacao_df %>%
  knitr::kable(caption = "Métricas de qualidade nos dados de validação") %>%
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

Tabela 7: Métricas de qualidade nos dados de validação

Modelo	Acuracia	Sensibilidade	Especificidade
Reg. Logística	45.99	86.61	27.28
Random Forest	56.98	70.33	50.83
SVM	65.00	51.00	71.00
RNA	55.00	83.00	36.00

Tabela 8: Métricas de qualidade nos dados de treino

Modelo	Acuracia	Sensibilidade	Especificidade
Reg. Logística	36.47	87.89	18.65
Random Forest	50.75	67.34	44.99
SVM	48.00	35.00	67.00
RNA	0.00	0.00	0.00

E, por fim, nos dados de teste, temos:

```
teste_df <- data.frame(Modelo = c("Reg. Logística","Random Forest","SVM","RNA"),
                       Acuracia = c(36.47,50.75,48.00,0),
                       Sensibilidade = c(87.89,67.34,35.00,0),
                       Especificidade = c(18.65,44.99,67.00,0))

teste_df %>%
  knitr::kable(caption = "Métricas de qualidade nos dados de treino") %>%
  kableExtra::kable_styling() %>%
  kableExtra::kable_classic_2(full_width = F)
```

Conclusão

Os resultados foram satisfatórios no sentido de as métricas analisadas serem semelhantes para o banco de treino e de validação, mesmo que estes sejam separados em ordem temporal. Isso é um indício de que o modelo, apesar de somente poder ser ajustado com dados do presente, deve poder ser utilizado para inferências futuras.

As métricas avaliadas mostraram-se relativamente próximas se comparadas entre os quatro modelos implementados, e nos casos possíveis, o ponto de corte mostrou-se importante para controlar a taxa de erros do tipo 1.

A classe de modelos que mostrou-se mais adequada para a resolução do problema foi a de Regressão Logística, sendo que os resultados obtidos na nossa implementação foram muito semelhantes aos da RNA, que mostrou-se ligeiramente superior quanto ao desempenho na base de dados. Além disso, a regressão logística permite uma alta interpretabilidade de seus parâmetros, o que é uma grande vantagem para um estudo mais estruturado das associações dos fatores contabilizados na modelagem com a variável resposta (contaminação de Covid-19). Essa classe de modelos costuma possuir um menor tempo de ajuste se comparados a modelos mais complexos, e também permite o controle de taxa de erro do tipo 1 por vários métodos disponíveis, inclusive considerando pesos para cada tipo de erro para a obtenção do ponto de corte ideal, que é um dos parâmetros de inferência mais importantes em problemas que envolvem saúde pública.

Links

- Repositório no GitHub
- OpenDataSus
- Apresentação no google docs
- Informações de treino da RNA em Comet.ml