Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Previsão da Eficiência de Extintores de Incêndio com Machine Learning

Conceituação e Definição do Problema de Negócio

O teste hidrostático do extintor é um procedimento estabelecido pelas normas ABNT NBR 12962/2016, que determinam que todos os extintores devem ser testados a cada cinco anos, com a finalidade de identificar eventuais vazamentos, além de também verificar a resistência do material do extintor.

Com isso, o teste hidrostático extintor pode ser realizado em baixa e alta pressão, de acordo com estas normas em questão. O procedimento é realizado por profissionais técnicos da área e com a utilização de aparelhos específicos e apropriados para o teste, visto que eles devem fornecer resultados com exatidão.

Seria possível usar Machine Learning para prever o funcionamento de um extintor de incêndio com base em simulações feitas em computador e assim incluir uma camada adicional de segurança nas operações de uma empresa? Esse é o objetivo deste projeto.

Usando dados reais disponíveis publicamente, nosso desafio é construir um modelo de Machine Learning capaz de prever se a chama será extinta ou não ao usar um extintor de incêndio.

O link abaixo contem os dados:

https://www.muratkoklu.com/datasets/vtdhnd07.php

Dicionário de Dados

O conjunto de dados foi obtido como resultado dos testes de extinção de quatro chamas de combustível diferentes com um sistema de ondas sonoras. O sistema de extinção de incêndio por ondas sonoras consiste em 4 subwoofers com uma potência total de 4.000 Watts. Existem dois amplificadores que permitem que o som chegue a esses subwoofers como amplificado.

A fonte de alimentação do sistema e o circuito do filtro garantem que a frequência de som seja transmitida adequadamente para o sistema localizado dentro da unidade de controle. Enquanto o computador é usado como fonte de frequência, o anemômetro é usado para medir o fluxo de ar resultante das ondas sonoras durante a fase de extinção da chama e um decibelímetro é usado para medir a intensidade do som.

Um termômetro infravermelho é utilizado para medir a temperatura da chama e da lata de combustível, e uma câmera é instalada para detectar o tempo de extinção da chama.

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Um total de 17.442 testes foram realizados com esta configuração experimental. Os experimentos foram planejados da seguinte forma:

- 3 diferentes combustíveis líquidos e 2 combustíveis GLP foram usados para criar a chama.
- 5 tamanhos diferentes de latas de combustível líquido foram usados para atingir diferentes tamanhos de chamas.
- O ajuste de meio cheio e completamente cheio de gás foi usado para o combustível GLP.

Durante a realização de cada experimento, o recipiente de combustível, a 10 cm de distância, foi movido para frente até 190 cm, aumentando a distância em 10 cm a cada vez. Junto com o recipiente de combustível, o anemômetro e o decibelímetro foram movidos para frente nas mesmas dimensões.

Experimentos de extinção de incêndio foram conduzidos com 54 ondas sonoras de frequências diferentes em cada distância e tamanha de chama.

Ao longo dos experimentos de extinção de chama, os dados obtidos de cada dispositivo de medição foram registrados e em conjunto de dados foi criado. O conjunto de dados inclui as características do tamanho do recipiente de combustível representando o tamanho da chama, tipo de combustível, frequência, decibéis, distância, fluxo de ar e extinção de chama. Assim, 6 recursos de entrada e 1 recurso de saída serão usados no modelo que vamos construir, inicialmente.

A coluna de Status (extinção de chama ou não extinção de chama) pode ser prevista usando os seis recursos de entrada no conjunto de dados. Os recursos de status e combustível são categóricos, enquanto outros recursos são numéricos.

Nosso desafio é construir um modelo de Machine Learning capaz de prever, com base em novos dados, se a chama será extinta ou não ao usar um extintor de incêndio.

Propriedades e Descrições dos Combustíveis Líquidos

Item	Valores	Unidade	Descrição
Tamanho	7, 12, 14, 16, 20	cm	Label Encoding → 7cm = 1, 12cm = 2, 14cm = 3, 16cm = 4, 20cm = 5
Combustivel	Gasolina, Querosene, Thiner	-	Tipos de Combustivel
Distância	10 - 190	cm	-
Decibeis	72 - 113	dB	-
Fluxo de Ar	0 - 17	m/s	-
Frequência	1 - 75	Hz	-
Status	0, 1	-	Label Encoding → 0 = Não Extinto, 1 = Extinto

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Propriedades e Descrição do GLP

Item	Valores	Unidade	Descrição
Tamanho	Válvula Meio-Aberta Válvulo Totalmente Aberta	-	Label Encoding → Meio-Aberta = 6 Totalmente Aberta = 7
Combustivel	LPG	-	Tipos de Combustivel
Distância	10 - 190	cm	-
Decibeis	72 - 113	dB	-
Fluxo de Ar	0 - 17	m/s	-
Frequência	1 - 75	Hz	-
Status	0, 1	-	Label Encoding → 0 = Não Extinto, 1 = Extinto

Pacotes de Trabalho

```
# Conferindo Diretório de Trabalho
    getwd()

  # Carregando Pacotes
    require(dplyr)
    require(ggplot2)
    require(gmodels)
    require(plotly)
    require(caret)
    require(readxl)
    require(randomForest)
    require(ROCR)
    require(pROC)
    require(ROSE)
```

Carregando os Dados

```
# Carregando os Dados
    Dados00 <-  read_xlsx('dados/Acoustic_Extinguisher_Fire_Dataset.xlsx',
                          sheet = 'A_E_Fire_Dataset')
    View(Dados00)
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

	SIZE	FUEL	DISTANCE	DESIBEL	AIRFLOW	FREQUENCY	STATUS
1	1	gasoline	10	96	0.0	75	0
2	1	gasoline	10	96	0.0	72	1
3	1	gasoline	10	96	2.6	70	1
4	1	gasoline	10	96	3.2	68	1
5	1	gasoline	10	109	4.5	67	1
6	1	gasoline	10	109	7.8	66	1
7	1	gasoline	10	103	9.7	65	1
8	1	gasoline	10	95	12.0	60	1
9	1	gasoline	10	102	13.3	55	1
10	1	gasoline	10	93	15.4	52	1
11	1	gasoline	10	93	15.1	51	1
12	1	gasoline	10	95	15.2	50	1
13	1	gasoline	10	110	15.4	48	1
14	1	gasoline	10	111	15.2	47	1
15	1	gasoline	10	109	15.4	46	1
16	1	gasoline	10	105	15.2	45	1
17	1	gasoline	10	111	16.0	44	1
18	1	gasoline	10	110	15.7	42	1
19	1	gasoline	10	106	15.4	40	1
20	1	gasoline	10	111	15.5	38	1
21	1	gasoline	10	110	15.2	36	1

```
str(Dados00)
```

```
>     str(Dados00)
tibble [17,442 × 7] (S3: tbl_df/tbl/data.frame)
 $ SIZE     : num [1:17442] 1 1 1 1 1 1 1 1 1 1 ...
 $ FUEL     : chr [1:17442] "gasoline" "gasoline" "gasoline" "gasoline" ...
 $ DISTANCE : num [1:17442] 10 10 10 10 10 10 10 10 10 10 ...
 $ DESIBEL  : num [1:17442] 96 96 96 96 109 109 103 95 102 93 ...
 $ AIRFLOW  : num [1:17442] 0 0 2.6 3.2 4.5 7.8 9.7 12 13.3 15.4 ...
 $ FREQUENCY: num [1:17442] 75 72 70 68 67 66 65 60 55 52 ...
 $ STATUS   : num [1:17442] 0 1 1 1 1 1 1 1 1 1 ...
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

```
dim(Dados00)
```

```
>     dim(Dados00)
[1] 17442     7
```

```
colnames(Dados00)
```

```
>     colnames(Dados00)
[1] "SIZE"      "FUEL"      "DISTANCE"  "DESIBEL"   "AIRFLOW"   "FREQUENCY" "STATUS"
```

Organização e Transformação dos Dados

```
# Dados Missing
      colSums(is.na(Dados00))
      # Não temos dados faltantes neste Dataset
```

```
>       colSums(is.na(Dados00))
     SIZE      FUEL  DISTANCE   DESIBEL   AIRFLOW FREQUENCY    STATUS
        0         0         0         0         0         0         0
>       # Não temos dados faltantes neste Dataset
```

```
# Trasnformando Variáveis para Tipo Fator
      Dados01 <- Dados00
      Dados01$FUEL <- as.factor(Dados01$FUEL)
      Dados01$STATUS <- as.factor(Dados01$STATUS)
      str(Dados01)
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

```
>       str(Dados01)
tibble [17,442 × 7] (S3: tbl_df/tbl/data.frame)
 $ SIZE     : num [1:17442] 1 1 1 1 1 1 1 1 1 1 ...
 $ FUEL     : Factor w/ 4 levels "gasoline","kerosene",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ DISTANCE : num [1:17442] 10 10 10 10 10 10 10 10 10 10 ...
 $ DESIBEL  : num [1:17442] 96 96 96 96 109 109 103 95 102 93 ...
 $ AIRFLOW  : num [1:17442] 0 0 2.6 3.2 4.5 7.8 9.7 12 13.3 15.4 ...
 $ FREQUENCY: num [1:17442] 75 72 70 68 67 66 65 60 55 52 ...
 $ STATUS   : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 2 2 2 2 ...
```

```
# Verificando o Balanceamento da Variavel Resposta
      round(prop.table(table(Dados01$STATUS)) * 100, digits = 2)
      # Temos dados balanceados, não precisando de qualquer técnica de imputação
```

```
    0     1
50.22 49.78
>       # Temos dados balanceados, não precisando de qualquer técnica de imputação
```

Exploração e Análise dos Dados

Exploração Geral

```
# Explorando os Dados
    summary(Dados01)
```

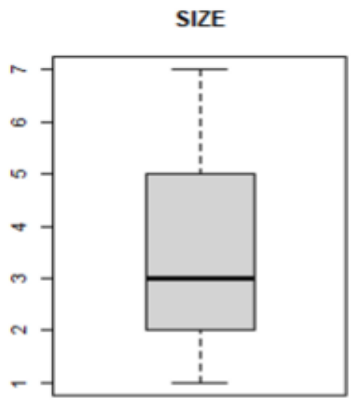
```
>     summary(Dados01)
      SIZE            FUEL        DISTANCE      DESIBEL         AIRFLOW         FREQUENCY      STATUS
 Min.   :1.000   gasoline:5130   Min.   : 10   Min.   : 72.00   Min.   : 0.000   Min.   : 1.00   0:8759
 1st Qu.:2.000   kerosene:5130   1st Qu.: 50   1st Qu.: 90.00   1st Qu.: 3.200   1st Qu.:14.00   1:8683
 Median :3.000   lpg     :2052   Median :100   Median : 95.00   Median : 5.800   Median :27.50
 Mean   :3.412   thinner :5130   Mean   :100   Mean   : 96.38   Mean   : 6.976   Mean   :31.61
 3rd Qu.:5.000                   3rd Qu.:150   3rd Qu.:104.00   3rd Qu.:11.200   3rd Qu.:47.00
 Max.   :7.000                   Max.   :190   Max.   :113.00   Max.   :17.000   Max.   :75.00
```

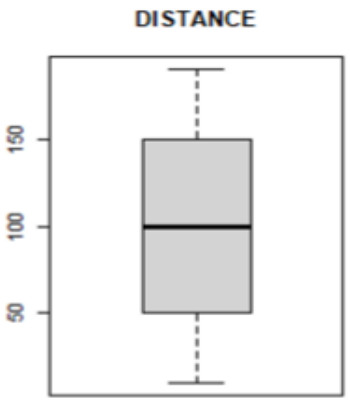
Podemos perceber que a variável categórica *STATUS* está balanceada, contendo volume equivalente de dados para ambas as respostas possíveis.

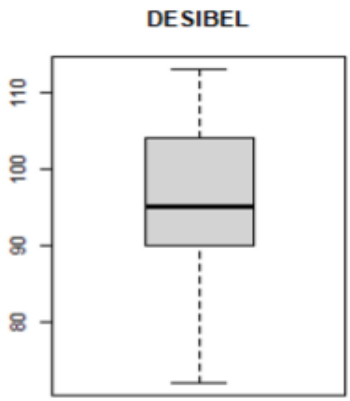
Por outro lado, a variável categórica *FUEL*, possui menor quantidade de dados sobre a categoria LPG. Vamos deixar aqui um ponto de oportunidade para uma possível revisão do modelo, caso seja necessário um aumento de performance preditiva.

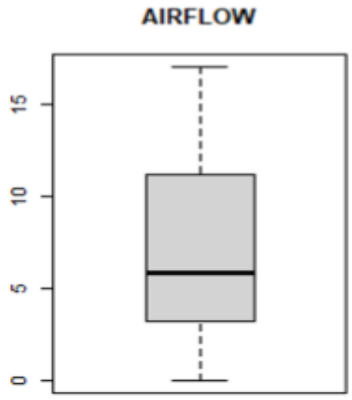
Analisando Variáveis Numéricas

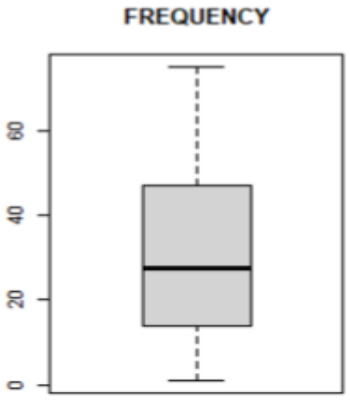
```
# Analisando Variáveis Numéricas

    i <- list(SIZE = 'SIZE', DISTANCE = 'DISTANCE', DESIBEL = 'DESIBEL',
              AIRFLOW = 'AIRFLOW', FREQUENCY = 'FREQUENCY')
    par(mfrow = c(2,3))
    for (x in i){
      boxplot(Dados01[x])
      title(x)
    }
```





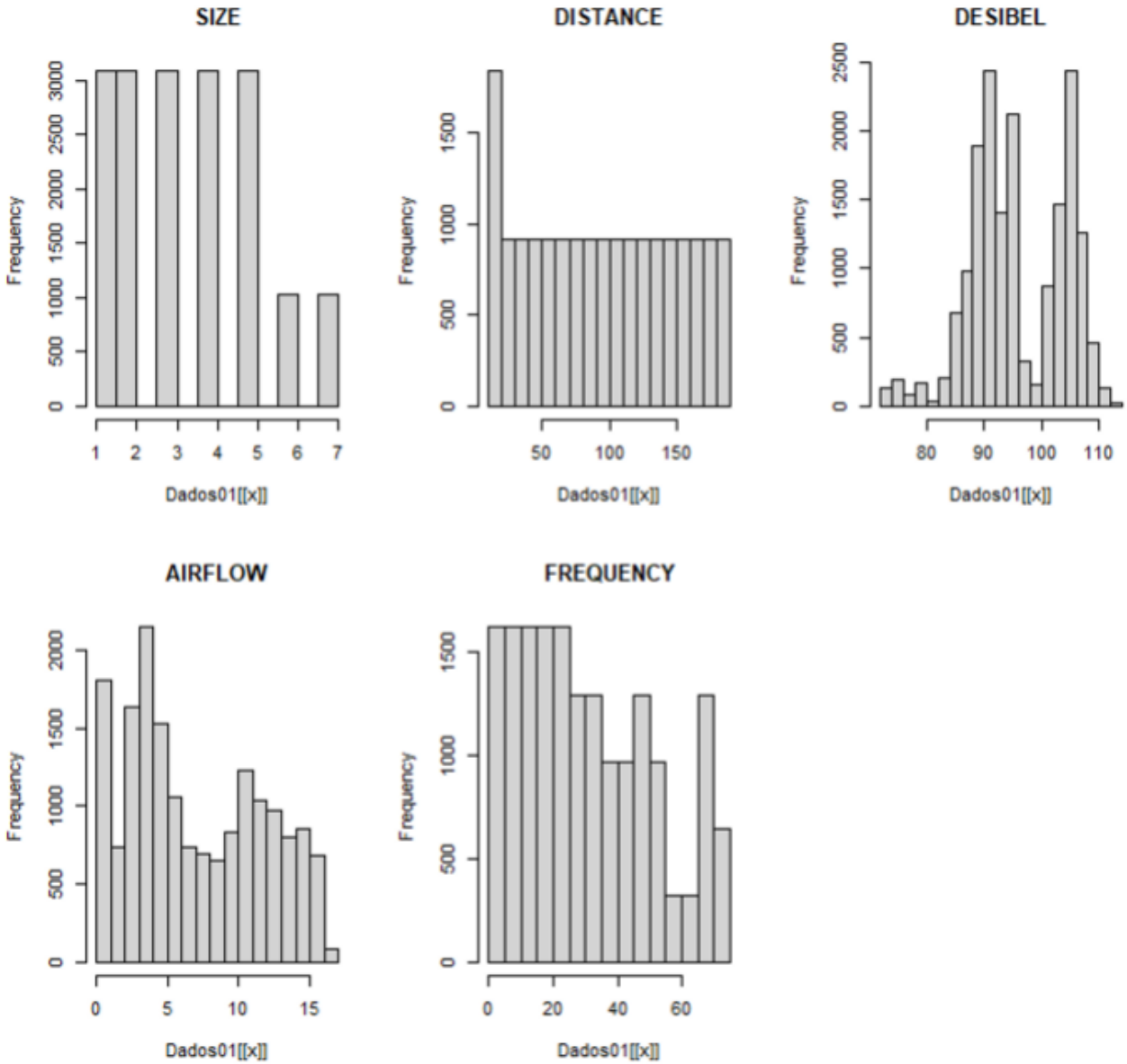






Analisando os gráficos de Boxplot para cada variável numérica, percebemos que não temos dados outliers que possam prejudicar o nosso modelo preditivo, portanto nenhuma técnica para tratamento de outliers é necessária neste caso.

```
par(mfrow = c(2,3))
for (x in i){
   hist(Dados01[[x]], main = x)
    }
```



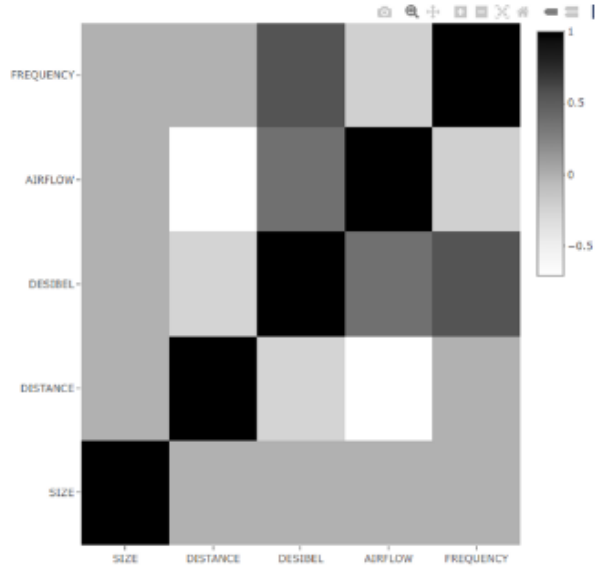
Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

A distribuição dos dados das variáveis numéricas não se caracteriza como um Padrão Gaussiano, porém poderíamos aplicar técnicas para aproximar a uma distribuição normal, caso fosse necessário atender alguma premissa de um teste de hipótese paramétrico por exemplo.

```
indicenum <- c(1, 3, 4, 5, 6)
DadosNum <- select(Dados01, all_of(indicenum))
View(DadosNum)
cor(DadosNum)
plot_ly(z = cor(DadosNum), type = "heatmap", colors = "Greys",
        x = colnames(DadosNum), y = colnames(DadosNum))
```

```
> cor(DadosNum)
                SIZE      DISTANCE       DESIBEL       AIRFLOW     FREQUENCY
SIZE       1.000000e+00 -1.115051e-20  1.619213e-19  6.852931e-19  1.964726e-19
DISTANCE  -1.115051e-20  1.000000e+00 -2.385897e-01 -7.074906e-01  0.000000e+00
DESIBEL    1.619213e-19 -2.385897e-01  1.000000e+00  3.769153e-01  5.617409e-01
AIRFLOW    6.852931e-19 -7.074906e-01  3.769153e-01  1.000000e+00 -2.120462e-01
FREQUENCY  1.964726e-19  0.000000e+00  5.617409e-01 -2.120462e-01  1.000000e+00
```



Algumas variáveis possuem correlação negativa alta entre si, é o caso de *AIRFLOW* x *DISTANCE* e outras, possui correlação alta positiva como *FREQUENCY* x *DECIBEL.* Talvez tenhamos que retirar algumas variáveis do processo para evitar multicolinearidade e baixa generalização do modelo preditivo.

Vamos trabalhar este tema mais a frente quando definirmos quais variáveis serão utilizadas no modelo preditivo base.

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Analisando Variáveis Categóricas

```
indiceCat <- c(2, 7)
DadosCat <- select(Dados01, all_of(indiceCat))
DadosCat
```

```
# Confirmando as Proporções de Cada Variável Categórica
        round(prop.table(table(DadosCat$FUEL)) * 100,digits = 2)
        round(prop.table(table(DadosCat$STATUS)) * 100, digits = 2)
```

```
gasoline kerosene      lpg  thinner 
   29.41    29.41    11.76    29.41 
```

```
    0     1 
50.22 49.78 
```

Podemos aferir que a variável *STATUS* (0 → 50,22%, 1 → 49,78%) está equilibrada, porém podemos ver que a variável *FUEL* possui menos observações experimentais com LPG (11,76% para 29,41% demais combustíveis).

Vamos confirmar o balanceamento com uma CrossTable:

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|

Total Observations in Table:  17442
```

DadosCat$FUEL	DadosCat$STATUS 0	1	Row Total
gasoline	2381	2749	5130
	14.787	14.916	
	0.464	0.536	0.294
	0.272	0.317	
	0.137	0.158	
kerosene	2831	2299	5130
	25.206	25.427	
	0.552	0.448	0.294
	0.323	0.265	
	0.162	0.132	

lpg	905	1147	2052
	15.277	15.411	
	0.441	0.559	0.118
	0.103	0.132	
	0.052	0.066	
thinner	2642	2488	5130
	1.682	1.697	
	0.515	0.485	0.294
	0.302	0.287	
	0.151	0.143	
Column Total	8759	8683	17442
	0.502	0.498	

Como já descrevemos anteriormente, temos um leve desbalanceamento para a categoria LPG, porém todo o dataset está equilibrado em relação as frequências relativas e absolutas de cada categoria.

Vamos deixar aqui outro ponto de possível revisão para aumento de performance, já que poderíamos aplicar alguma técnica de imputação para estes dados e equilibrar mais ainda o Dataset.

Para finalizar esta etapa, vamos aplicar um Teste Qui quadrado para verificar se as duas variáveis categóricas se comportam de forma semelhante quanto a dispersão de seus dados.

Teste Qui quadrado

- Hipótese H0 → Não há relação entre *FUEL* e *STATUS*
- Hipótese H1 → *FUEL* e *STATUS* estão relacionadas

Se p-value for menos que 0.05, rejeitamos H0.

```
chisq.test(table(DadosCat$FUEL, DadosCat$STATUS))
```

```
        Pearson's Chi-squared test

data:  table(DadosCat$FUEL, DadosCat$STATUS)
X-squared = 114.4, df = 3, p-value < 2.2e-16
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

Pré-processamento dos Dados

Criando Datasets de Treino e Teste

```
set.seed(10)
Partition <- createDataPartition(y = Dados01$STATUS, p = 0.75, list = FALSE)
DadosTreino <- Dados01[Partition,]
DadosTeste <- Dados01[-Partition,]
```

Normalizando os Dados

Temos dados de diferentes escalas, portanto vamos utilizar a função scale() para que não tenhamos influência de magnitude de uma variável no modelo preditivo.

```
DadosTreinoNumNorm <- scale(select(DadosTreino, all_of(indicenum)))

DadosTesteNumNorm <- scale(select(DadosTeste, all_of(indicenum)))

DadosTreinoCat <- select(DadosTreino, all_of(indiceCat))

DadosTesteCat <- select(DadosTeste, all_of(indiceCat))

# Novos Datasets Normalizados

DadosTreinoNorm <- cbind(DadosTreinoNumNorm, DadosTreinoCat)

DadosTesteNorm <- cbind(DadosTesteNumNorm, DadosTesteCat)

DadosNorm <- rbind(DadosTreinoNorm, DadosTesteNorm)

View(DadosNorm)
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

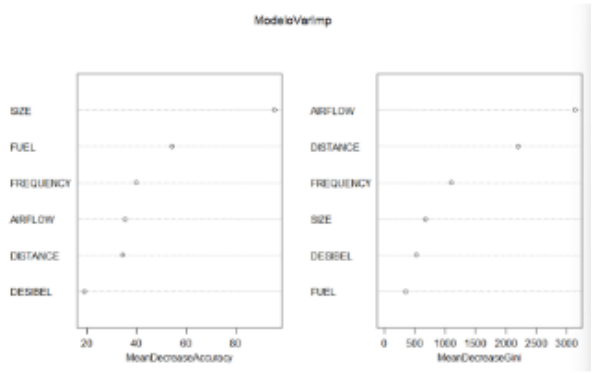
	SIZE	DISTANCE	DESIBEL	AIRFLOW	FREQUENCY	FUEL	STATUS
1	-1.380091	-1.6421635	-0.04106666	-1.472595224	2.07503092	gasoline	0
2	-1.380091	-1.6421635	-0.04106666	-0.923548550	1.83624491	gasoline	1
3	-1.380091	-1.6421635	-0.04106666	-0.796845471	1.74073051	gasoline	1
4	-1.380091	-1.6421635	1.53951177	-0.522322135	1.69297330	gasoline	1
5	-1.380091	-1.6421635	1.53951177	0.174544797	1.64521610	gasoline	1
6	-1.380091	-1.6421635	0.81001403	0.575771213	1.59745890	gasoline	1
7	-1.380091	-1.6421635	-0.16264961	1.061466347	1.35867288	gasoline	1
8	-1.380091	-1.6421635	0.68843108	1.335989684	1.11988687	gasoline	1
9	-1.380091	-1.6421635	-0.40581553	1.779450459	0.97661526	gasoline	1
10	-1.380091	-1.6421635	-0.40581553	1.716098920	0.92885806	gasoline	1
11	-1.380091	-1.6421635	-0.16264961	1.737216099	0.88110086	gasoline	1
12	-1.380091	-1.6421635	1.66109472	1.779450459	0.78558645	gasoline	1
13	-1.380091	-1.6421635	1.53951177	1.779450459	0.69007205	gasoline	1
14	-1.380091	-1.6421635	1.05317995	1.737216099	0.64231485	gasoline	1
15	-1.380091	-1.6421635	1.78267768	1.906153537	0.59455764	gasoline	1
16	-1.380091	-1.6421635	1.66109472	1.842801998	0.49904324	gasoline	1
17	-1.380091	-1.6421635	1.17476290	1.779450459	0.40352883	gasoline	1
18	-1.380091	-1.6421635	1.78267768	1.800567639	0.30801443	gasoline	1
19	-1.380091	-1.6421635	1.66109472	1.737216099	0.21250002	gasoline	1
20	-1.380091	-1.6421635	0.81001403	1.673864560	0.16474282	gasoline	1
21	-1.380091	-1.6421635	1.53951177	1.673864560	0.11698562	gasoline	1
22	-1.380091	-1.6421635	1.41792881	1.673864560	0.06922841	gasoline	1
23	-1.380091	-1.6421635	1.66109472	1.716098920	0.02147121	gasoline	1
24	-1.380091	-1.6421635	1.53951177	2.117325335	-0.07404319	gasoline	1

Identificando as Variáveis de Maior Influência na Variável Resposta STATUS

Vamos utilizar o algoritmo de Random Forest para identificar quais são as variáveis de maior importância e desta forma tentar simplificar nossos modelos preditivos reduzindo a dimensionalidade.

```
ModeloVarImp <- randomForest(STATUS ~ ., data = DadosNorm, ntree = 100,
                             nodesize = 10, importance = T)

varImp(ModeloVarImp)
varImpPlot(ModeloVarImp)
```

```
>         varImp(ModeloVarImp)
                0        1
SIZE       76.30568 76.30568
DISTANCE   27.91490 27.91490
DESIBEL    13.67281 13.67281
AIRFLOW    28.32555 28.32555
FREQUENCY  29.36421 29.36421
FUEL       48.19618 48.19618
```



Inicialmente para nosso primeiro Modelo, vamos utilizar as variáveis com valores de Mean Accuracy acima de 25, sendo elas SIZE, FUEL, FREQUENCY, AIRFLOW e DISTANCE.

Modelagem Preditiva

Modelo Base → Random Forest (CARET)

```
# Criando e Treinando o Modelo00 (Base) de Previsão

        Modelo00 <- train(STATUS ~ SIZE + FUEL + FREQUENCY + AIRFLOW + DISTANCE,
                          data = DadosTreinoNorm, method = 'rf')
```

```
# Fazendo as Previsões com Dados de Teste

        Previsoes <- predict(Modelo00, newdata = DadosTesteNorm)
```

```
# Avaliando Performance do Modelo01
        mean(Previsoes==DadosTesteNorm$STATUS)
```

```
>         mean(Previsoes==DadosTesteNorm$STATUS)
[1] 0.973159
```

```
round(prop.table(table(Previsoes, DadosTesteNorm$STATUS)) * 100, digits = 2)
```

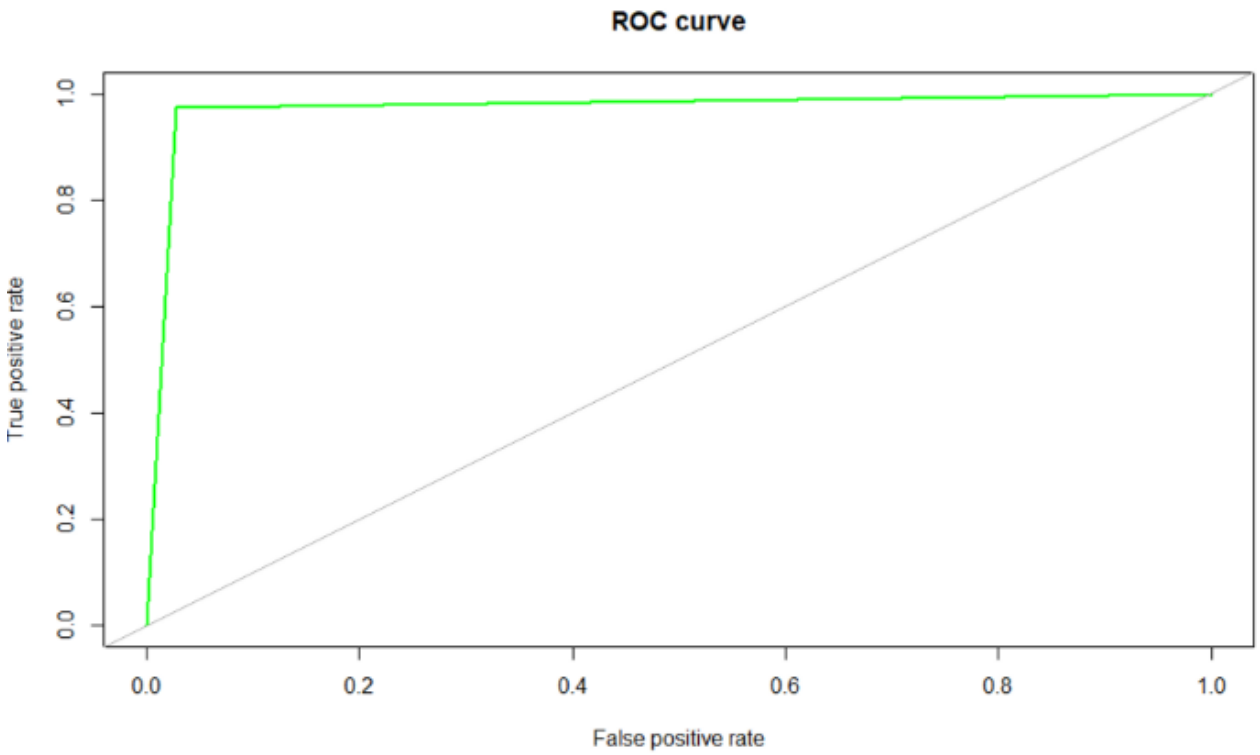
```
Previsoes     0     1
        0 48.82  1.28
        1  1.40 48.50
```

```
confusionMatrix(DadosTesteNorm$STATUS, Previsoes, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2128   61
         1   56 2114

               Accuracy : 0.9732
                 95% CI : (0.9679, 0.9778)
    No Information Rate : 0.501
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9463

 Mcnemar's Test P-Value : 0.7115

            Sensitivity : 0.9720
            Specificity : 0.9744
         Pos Pred Value : 0.9742
         Neg Pred Value : 0.9721
             Prevalence : 0.4990
         Detection Rate : 0.4850
   Detection Prevalence : 0.4978
      Balanced Accuracy : 0.9732

       'Positive' Class : 1
```

```
roc.curve(DadosTesteNorm$STATUS, Previsoes, plotit = T, col = "green",
              add.roc = FALSE)
```

Temos uma acurácia de 97,3% das previsões com o modelo proposto, resultado bastante satisfatório e comprová-lo pelas Métricas de Accuracy e ACU.



Modelo 02 → GLM (CARET)

```
# Criando e Treinando o Modelo02 de Previsão

        Modelo02 <- train(STATUS ~ SIZE + FUEL + FREQUENCY + AIRFLOW + DISTANCE,
                          data = DadosTreinoNorm, method = 'glm')
```

```
# Fazendo as Previsões com Dados de Teste

        Previsoes02 <- predict(Modelo02, newdata = DadosTesteNorm)
```

```
# Avaliando Performance do Modelo01
        mean(Previsoes02==DadosTesteNorm$STATUS)
```

```
[1] 0.9018123
```

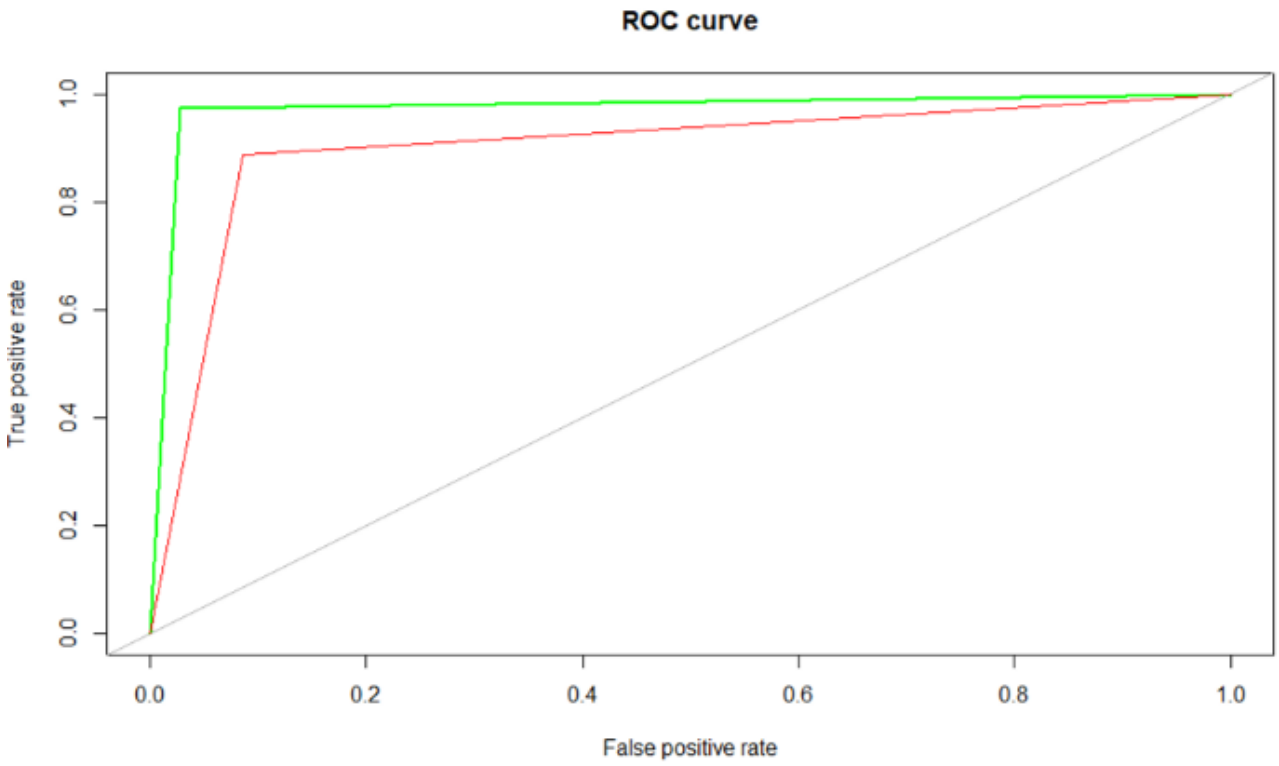
```
round(prop.table(table(Previsoes02, DadosTesteNorm$STATUS)) * 100, digits = 2)
```

```
Previsoes02     0     1
          0 45.93  5.53
          1  4.29 44.25
```

```
confusionMatrix(DadosTesteNorm$STATUS, Previsoes02, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2002  187
         1  241 1929

               Accuracy : 0.9018
                 95% CI : (0.8926, 0.9105)
    No Information Rate : 0.5146
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8036

 Mcnemar's Test P-Value : 0.01041

            Sensitivity : 0.9116
            Specificity : 0.8926
         Pos Pred Value : 0.8889
         Neg Pred Value : 0.9146
             Prevalence : 0.4854
         Detection Rate : 0.4425
   Detection Prevalence : 0.4978
      Balanced Accuracy : 0.9021

       'Positive' Class : 1
```

```
roc.curve(DadosTesteNorm$STATUS, Previsoes02, plotit = T, col = "red",
                  add.roc = TRUE)
```



Temos uma acurácia de 90,18% das previsões com o novo modelo, resultado inferior ao Modelo Base, comparando as métricas de Accuracy e ACU. Vamos construir mais um modelo, agora com Árvores de Decisão.

Modelo 03 → Árvores de Decisão RPART (CARET)

```
# Criando e Treinando o Modelo03 de Previsão

        Modelo03 <- train(STATUS ~ SIZE + FUEL + FREQUENCY + AIRFLOW + DISTANCE,
                          data = DadosTreinoNorm, method = 'rpart')
```

```
# Fazendo as Previsões com Dados de Teste

        Previsoes03 <- predict(Modelo03, newdata = DadosTesteNorm)
```

Elaborado por Thiago Bulgarelli

Contato: bugath36@gmail.com

```
# Avaliando Performance do Modelo01
        mean(Previsoes03==DadosTesteNorm$STATUS)
```

```
[1] 0.8873595
```

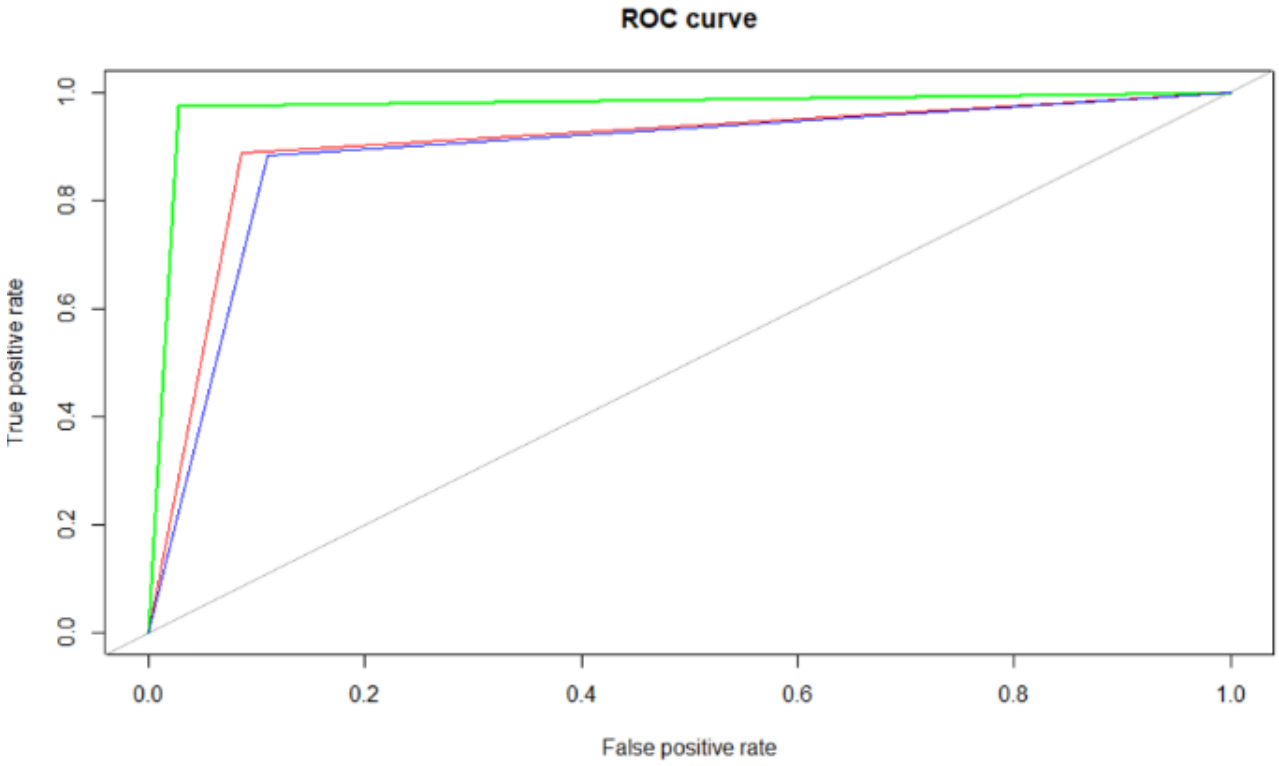
```
round(prop.table(table(Previsoes03, DadosTesteNorm$STATUS)) * 100, digits = 2)
```

```
Previsoes03     0     1
          0 44.71  5.76
          1  5.51 44.02
```

```
confusionMatrix(DadosTesteNorm$STATUS, Previsoes03, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1949  240
         1  251 1919

               Accuracy : 0.8874
                 95% CI : (0.8776, 0.8966)
    No Information Rate : 0.5047
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7747

 Mcnemar's Test P-Value : 0.6518

            Sensitivity : 0.8888
            Specificity : 0.8859
         Pos Pred Value : 0.8843
         Neg Pred Value : 0.8904
             Prevalence : 0.4953
         Detection Rate : 0.4402
   Detection Prevalence : 0.4978
      Balanced Accuracy : 0.8874

       'Positive' Class : 1
```

```
roc.curve(DadosTesteNorm$STATUS, Previsoes03, plotit = T, col = "blue",
                  add.roc = TRUE)
```



Para este modelo temos uma acurácia ainda menor, 88,7% das previsões com o modelo proposto.

Conclusão

Vamos constatar que nosso problema de negócio não exige uma acurácia muito alta, visto que a intenção é auxiliar o processo de detecção de eficiência de extintores, pois os testes laboratoriais devem ser feitos e refeitos por amostragens, conforme solicita a legislação e controle dos órgãos fiscalizadores.

Sendo assim, precisamos de um modelo que apesar de prever alguns dados errados (falsos positivos ou falsos negativos), consiga entregar uma previsibilidade generalizável capaz de auxiliar na execução dos testes experimentais, reduzindo tempo, custos e exposição a riscos dos executores.