Machine Learning na Segurança do Trabalho: Prevendo a Eficiência de Extintores de Incêndio

José Aleilson

Introdução

Um procedimento estabelecido pelas normas da ABNT é o teste hidrostático de extintor. Esse teste determina que todos os extintores devem ser testados a cada cinco anos com o objetivo de detectar possíveis falhas.

O teste hidrostático de extintor pode ser realizado em diversas formas de pressão. Esse procedimento é realizado por profissionais técnicos dessa área, que utilizam equipamentos e aparelhos específicos para esse teste.

No entanto, com o objetivo de aumentar a segurança nesse procedimento, surge a questão de se seria possível criar um modelo de Machine Learning capaz de **prever o funcionamento de um extintor de incêndio com base em simulações feitas no computador, adicionando assim mais uma camada de segurança a esse trabalho**.

Os dados obtidos neste trabalho são resultado de um experimento de extinção de chamas utilizando quatro tipos diferentes de combustíveis, utilizando um sistema de extinção por ondas sonoras. A partir desses dados, foram fornecidos seis recursos de entrada e um de saída, que será objeto de predição do nosso modelo de Machine Learning. Essa variável nos indica se o experimento foi bem-sucedido ou falhou, ou seja, se conseguimos extinguir as chamas para os respectivos combustíveis e situações durante o teste.

Metodologia

A metodologia adotada consiste em realizar uma análise exploratória dos dados, seguida pelo pré-processamento e modelagem de Machine Learning. Os modelos a serem explorados incluem GLM, KNN, Random Forest e Naive Bayes. A análise dos resultados envolverá comparação de desempenho dos modelos, além da visualização dos resultados com gráficos.

1. Análise exploratória de dados:

Realizar uma análise preliminar dos dados para compreender sua estrutura e características. Utilizar a biblioteca ggplot2 para criar visualizações gráficas que ajudem na compreensão dos dados. Explorar a distribuição das variáveis e identificar possíveis padrões ou outliers.

2. Pré-processamento dos dados:

Realizar limpeza dos dados, tratando valores ausentes, inconsistências e outliers, se necessário. Utilizar a biblioteca dplyr para realizar transformações nos dados, como filtragem, agregação ou criação de novas variáveis, se necessário. Separar os dados em conjuntos de treinamento e teste.

3. Modelagem de Machine Learning:

Utilizar a biblioteca e1071 para criar um modelo de Naive Bayes. Utilizar a biblioteca randomForest para criar um modelo de Random Forest. Utilizar a biblioteca caret para criar modelos adicionais, como o GLM e KNN, e realizar a otimização de hiperparâmetros. Treinar e avaliar os modelos

4. Análise dos resultados:

Comparar o desempenho dos diferentes modelos de Machine Learning utilizando métricas apropriadas, como acurácia, precisão, recall ou F1-score. Analisar as características e importância das variáveis no contexto do problema. Identificar o modelo com melhor desempenho para a previsão do funcionamento do extintor.

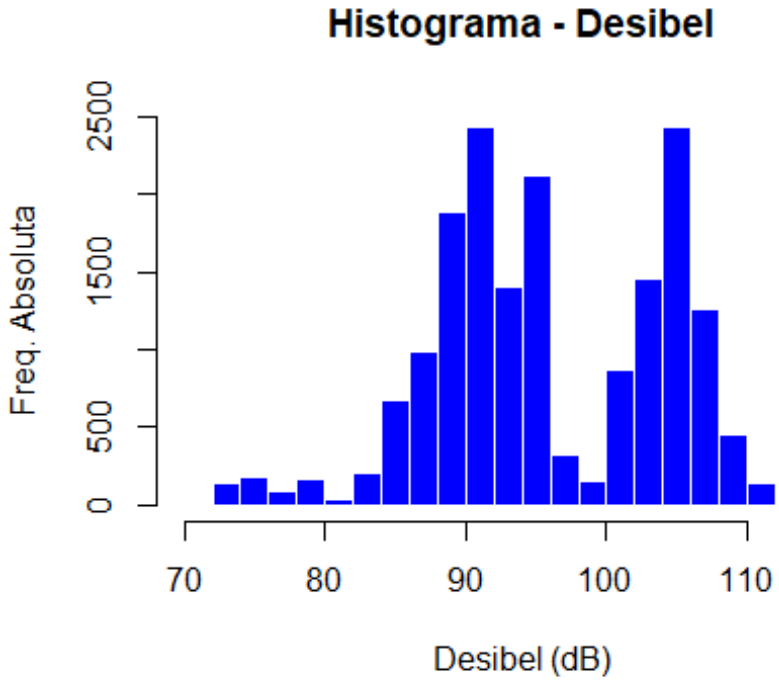
Análise Exploratória

Serão exploradas as distribuições das variáveis e identificados possíveis padrões ou outliers. Serão aplicadas técnicas de limpeza de dados e transformações. A análise exploratória tem como objetivo compreender a estrutura e características dos dados, fornecendo insights iniciais para o pré-processamento e a modelagem subsequente.

Análise Univariada

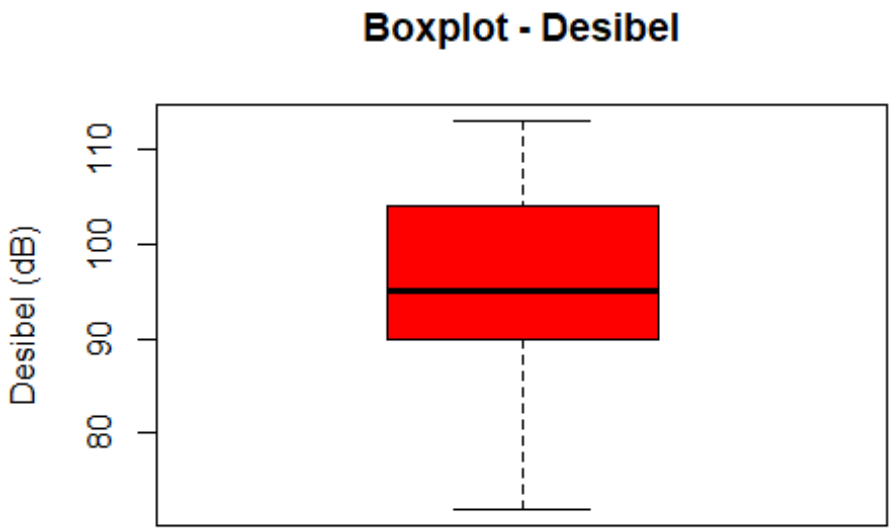
Desibel

A variável decibel contém informações sobre a intensidade do som. Nesse experimento de extinção de chamas, diferentes ondas sonoras foram utilizadas com o objetivo de extinguir as chamas. Para medir a intensidade do som, um decibelímetro foi utilizado.



Ao investigar o histograma resultante dos dados obtidos através do decibelímetro, cujo objetivo era medir a intensidade do som durante o experimento, percebe-se que a variável Decibel não segue uma distribuição normal. Além disso, esse vetor apresenta duas áreas de concentração durante a distribuição: uma no intervalo de 80 a 100 dB e outra de 100 a 113 dB.

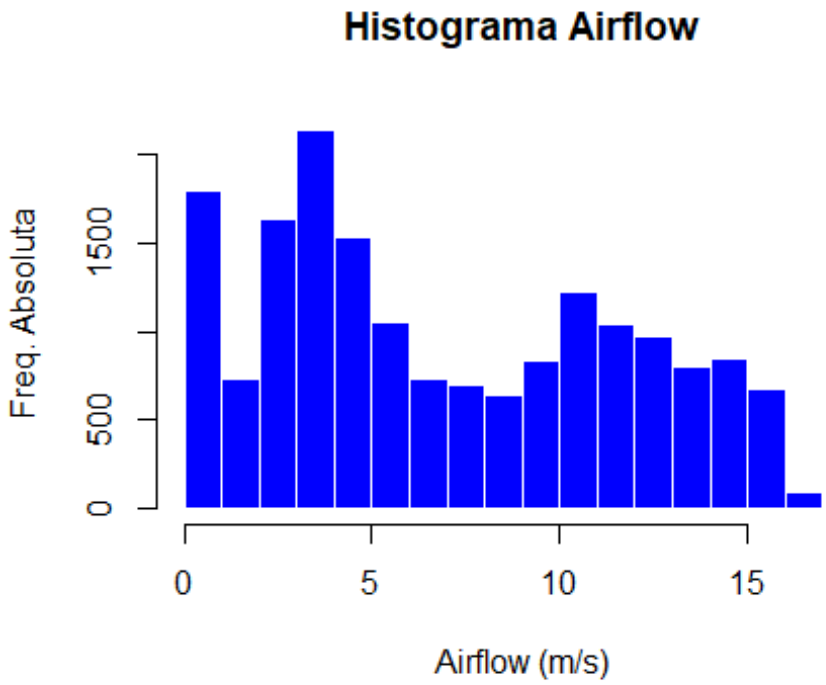
Agora vamos analisar um boxplot da variável decibel para identificar possíveis valores atípicos (outliers).



A variável decibel tem como valor mínimo 72 dB e valor máximo 113 dB. Essa variável apresentou uma maior concentração de dados no terceiro quartil em comparação com o primeiro quartil. Ao observar o boxplot do decibel, não foram detectados valores extremos.

Airflow

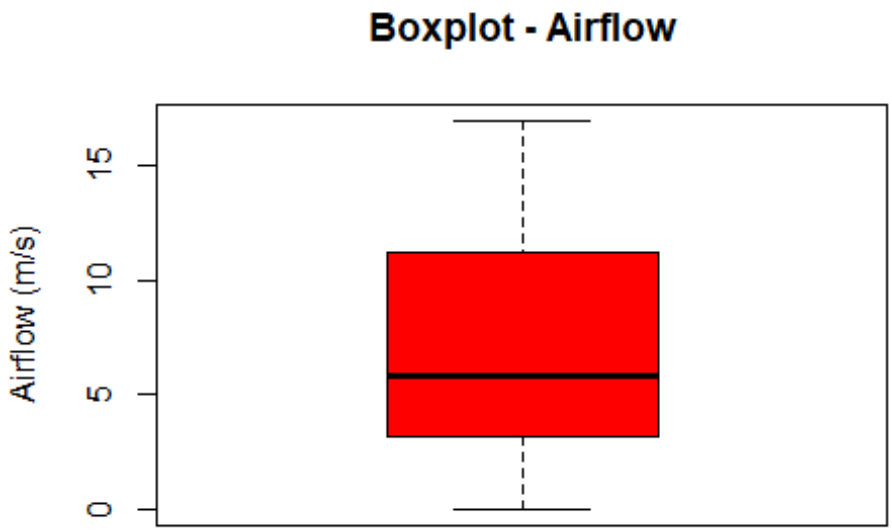
Os valores dessa variável foram obtidos durante a fase de extinção das chamas. Esses valores foram gerados pelas ondas sonoras utilizadas durante o processo de extinção, tanto pelo extintor de incêndio quanto pelo recipiente com a chama.



Observando o histograma da variável Airflow, nota-se que os maiores picos dos dados estão localizados no primeiro intervalo, com fluxo de ar de 0 a 5. A contagem mais alta de Fluxo de Ar é próxima a 5, enquanto a segunda contagem mais alta é próxima a 0, o que indica que os extintores de incêndio estão em estado de repouso.

Além disso, a variável “Airflow” não possui uma distribuição em formato de sino, o que indica que não segue uma distribuição normal.

Agora iremos analisar um boxplot da variável Airflow para identificar possíveis valores outliers.

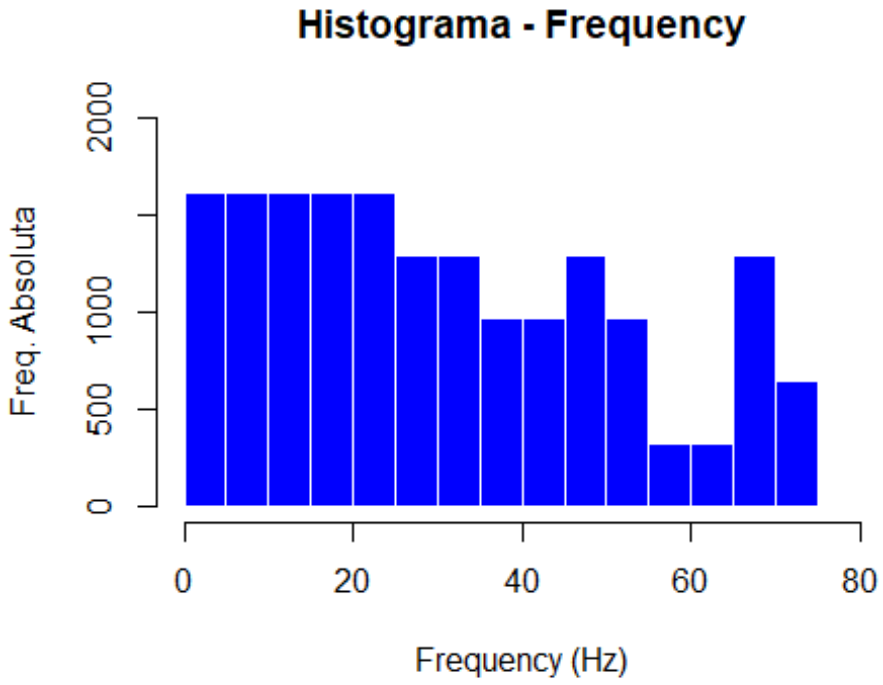


O limite inferior da variável “Airflow” foi de 0 m/s, indicando que esse fluxo de ar ocorre quando o experimento está em estado de repouso. Já o limite superior observado no boxplot foi de 17 m/s.

O valor do primeiro quartil foi de 3,2 m/s, enquanto a linha preta que representa a mediana (2º quartil) está localizada em 5,8 m/s e o terceiro quartil em 11,2 m/s. Em relação ao box plot, o terceiro quartil apresenta uma maior concentração dos dados, e nenhum valor considerado outlier foi encontrado.

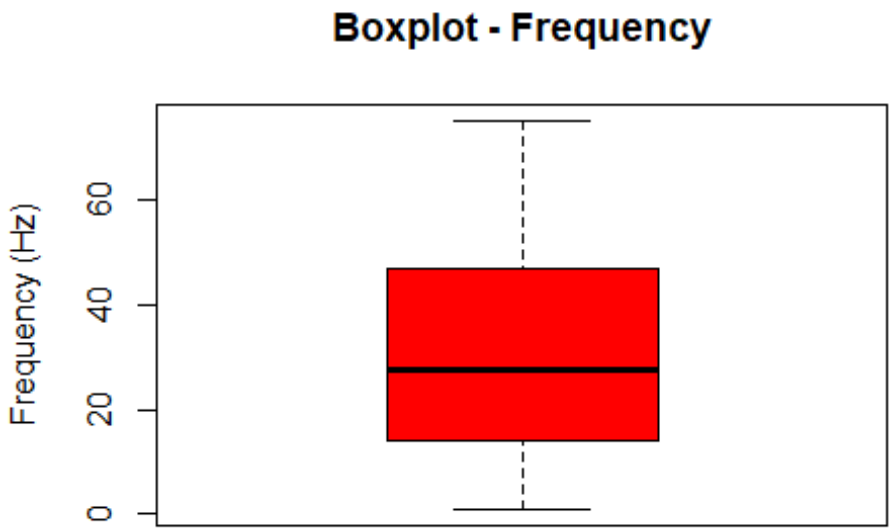
Frequency

Esse vetor representa a frequência das ondas sonoras no processo de extinção das chamas. Os experimentos de extinção de incêndio foram conduzidos utilizando 54 ondas sonoras de frequências diferentes para cada distância e tamanho da chama.



A frequência do som durante o experimento possui uma distribuição uniforme no intervalo de 0 a 20 Hz, seguida de uma queda no intervalo seguinte. Posteriormente, mantém-se de forma irregular nas medidas de frequência subsequentes.

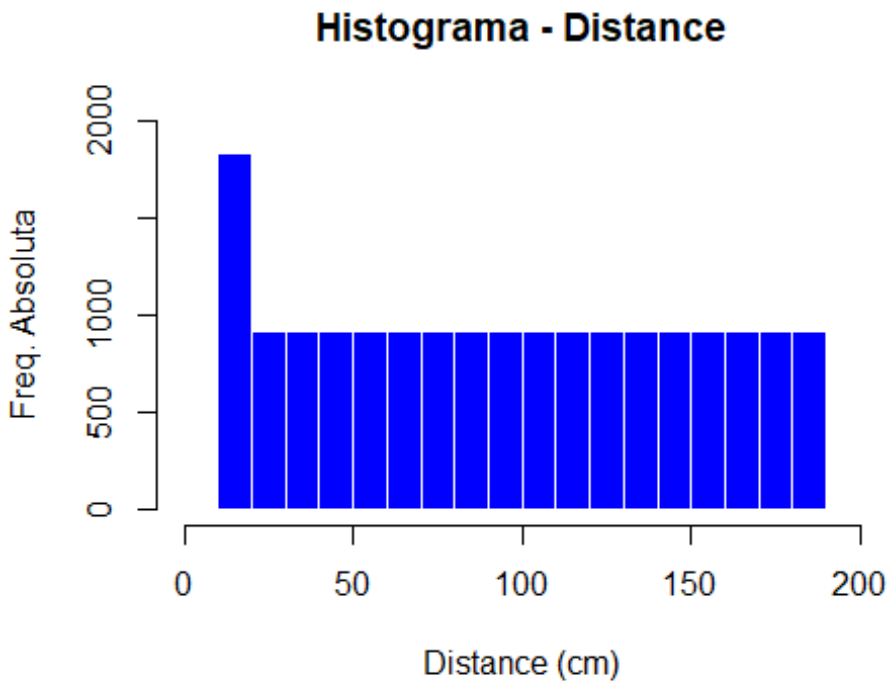
Vamos analisar um boxplot para verificar a existência de possíveis valores atípicos.



Os valores mínimos e máximos observados na variável frequency foram, respectivamente, 1 Hz e 75 Hz, enquanto a mediana, que representa o segundo quartil, foi de 27,5 Hz. Não foram detectados valores outliers nessa variável. Foi observada uma maior concentração de dados no terceiro quartil.

Distance

Essa variável representa a distância entre o extintor e o recipiente de combustível. Durante a realização de cada experimento, o recipiente de combustível, inicialmente posicionado a uma distância de 10 cm, foi movido para frente até atingir 190 cm, aumentando a distância em 10 cm a cada etapa.

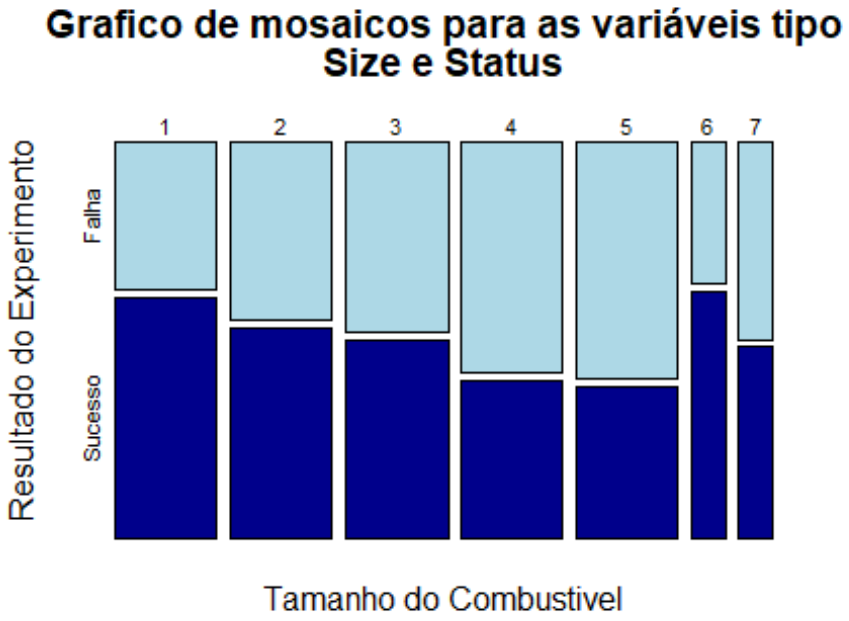


A única barra de contagem irregular no histograma é a primeira, que apresentou uma contagem de 1836. As demais barras seguem uma distribuição uniforme. Assim como observado nas variáveis anteriores, esta variável também não apresenta uma distribuição normal.

Análise Bivariada

Size e Status

O vetor Size traz informações relacionadas aos tamanhos dos recipientes de combustíveis. Durante o experimento, foram utilizados dois tipos de combustíveis: líquido e gás. Para os combustíveis líquidos, temos cinco tamanhos diferentes medidos em centímetros. Para os combustíveis a gás, temos a opção de ajuste de gás para meio e cheio.



Ao considerar os combustíveis líquidos, ao observar a relação entre as variáveis Size (que representa o tamanho do recipiente de combustível/tamanho da chama) e Status (que indica se o experimento de extinguir as chamas foi um sucesso ou não), nota-se que, à medida que o tamanho do recipiente aumenta de 1 a 5, ou seja, o tamanho da chama aumenta, o número de falhas ao extinguir a chama também aumenta. Consequentemente, o número de sucessos na extinção das chamas no experimento diminui.

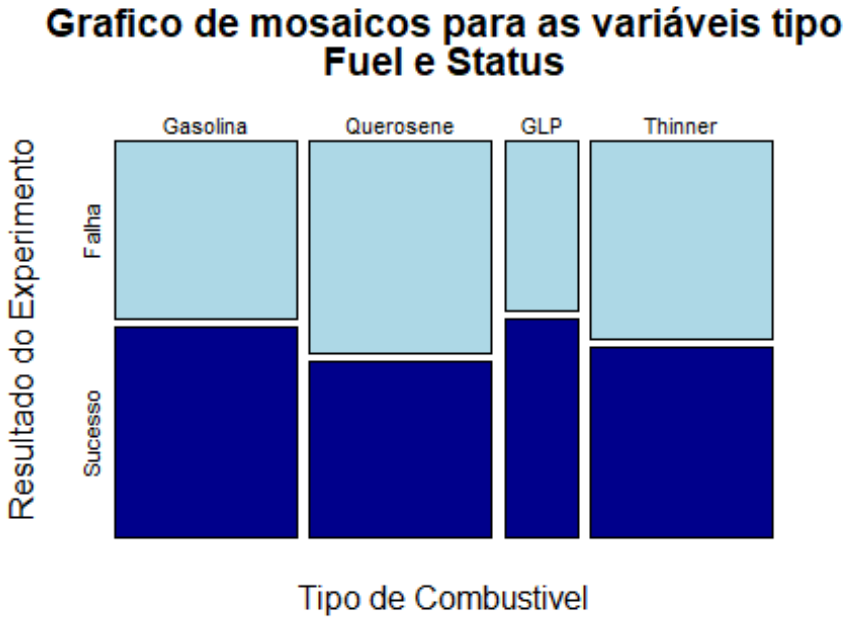
Considerando o combustível GLP (gás), observa-se que, à medida que o tamanho da chama aumenta de 6 a 7, ou seja, o recipiente muda de meio para cheio, o número de falhas ao extinguir as chamas também aumenta. Por outro lado, o número de sucessos diminui.

Com isso, é possível observar um padrão após analisar as informações obtidas a partir desse gráfico entre as variáveis Size e Status. Observa-se um padrão inverso,

em que, à medida que o tamanho do recipiente de combustível aumenta, o número de sucessos ao extinguir as chamas reduz.

Fuel e Status

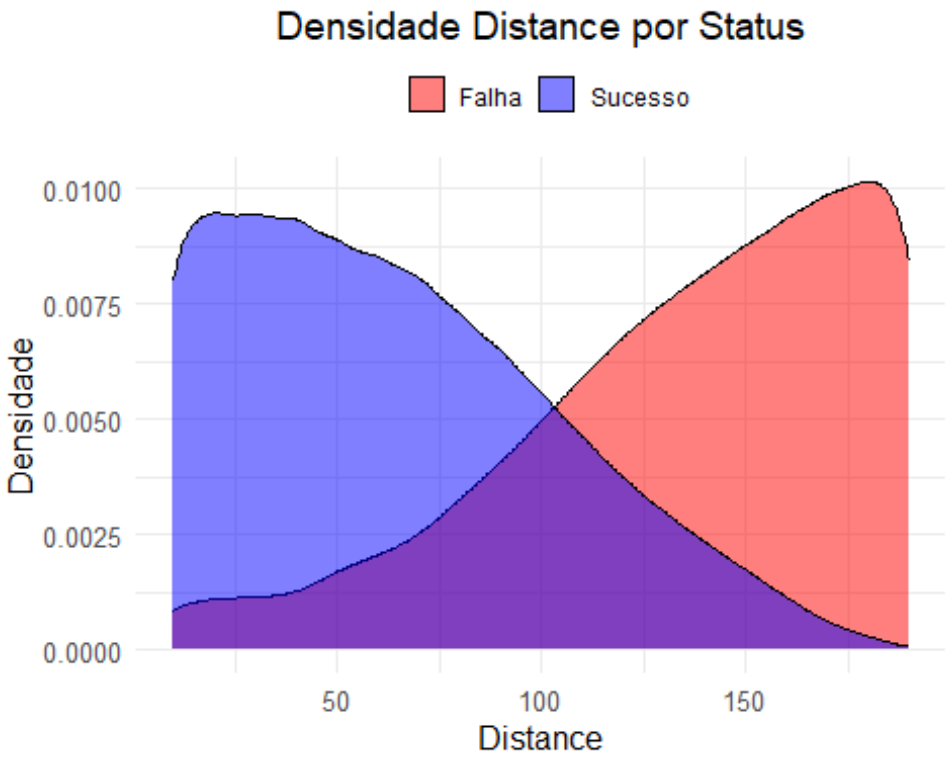
Nesse experimento, foram utilizados dois tipos de combustíveis: líquidos e combustíveis a gás. Entre os líquidos, temos três tipos distintos, enquanto para os combustíveis a gás temos dois tipos.



O número de sucessos observados no experimento variou entre os diferentes tipos de combustíveis. No caso da gasolina, foi observado um maior número de sucessos em comparação com as falhas relacionadas a esse combustível. Por outro lado, o querosene apresentou um número superior de falhas ao extinguir as chamas. Esse mesmo padrão se repete com o thinner Já o GLP (GÁS DE PETRÓLEO LIQUEFEITO) tem um comportamento semelhante ao da gasolina, com um número superior de sucessos na tentativa de extinguir as chamas.

Dintance e Status

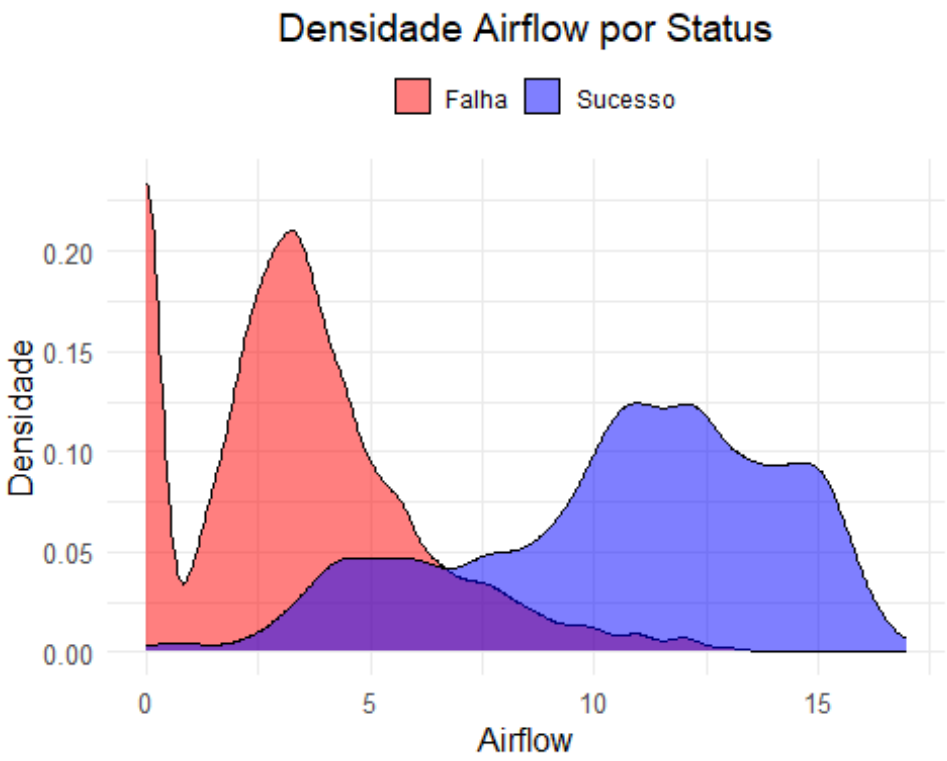
1. O que acontece com o experimento a medida que a distância aumenta?



Ao observar o gráfico acima, podemos deduzir que à medida que a distância aumenta, o número de sucessos na extinção de chamas diminui. Outro ponto interessante observado nesse gráfico é que quando a distância atinge o intervalo de 100 a 110 cm, as linhas de falhas e sucessos se cruzam, indicando que nessa distância as falhas e os sucessos são equilibrados.

Airflow e Status

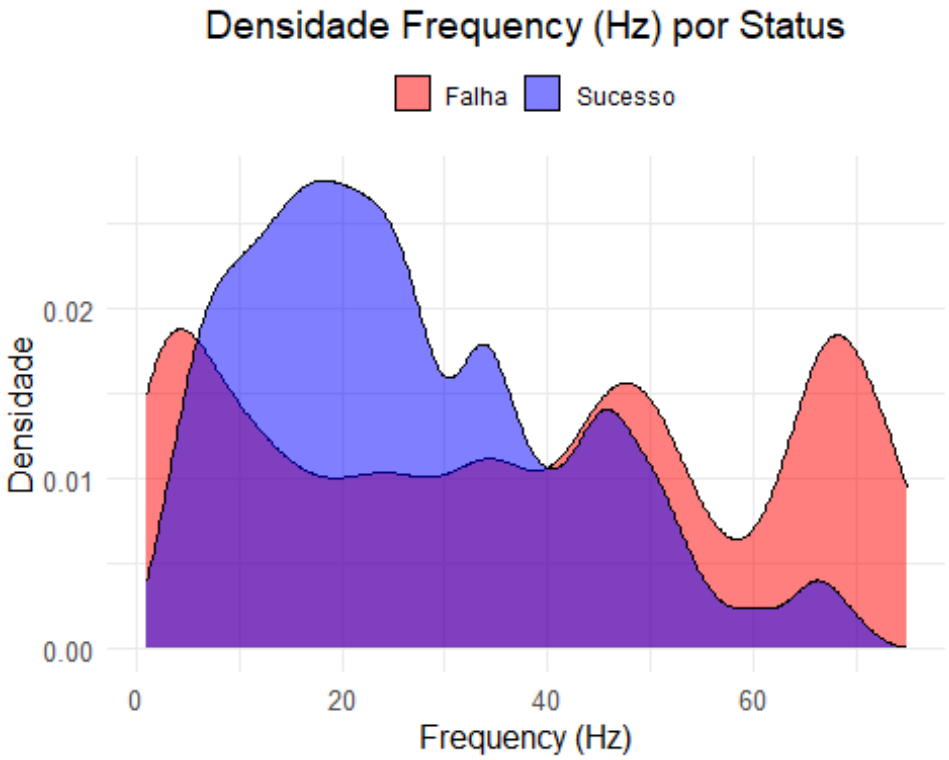
2. Como o Fluxo de Ar afeta o sucesso do experimento?



Ao observar o gráfico, especialmente as áreas em vermelho que representam as falhas, é possível perceber que a maioria delas está concentrada nos fluxos de ar mais baixos. Por outro lado, os fluxos de ar mais altos apresentam uma maior proporção de sucesso na extinção das chamas. Portanto, realizando uma análise simples desse gráfico, podemos concluir que quanto maior o fluxo de ar, maiores são as chances de sucesso no experimento.

Frequency e Status

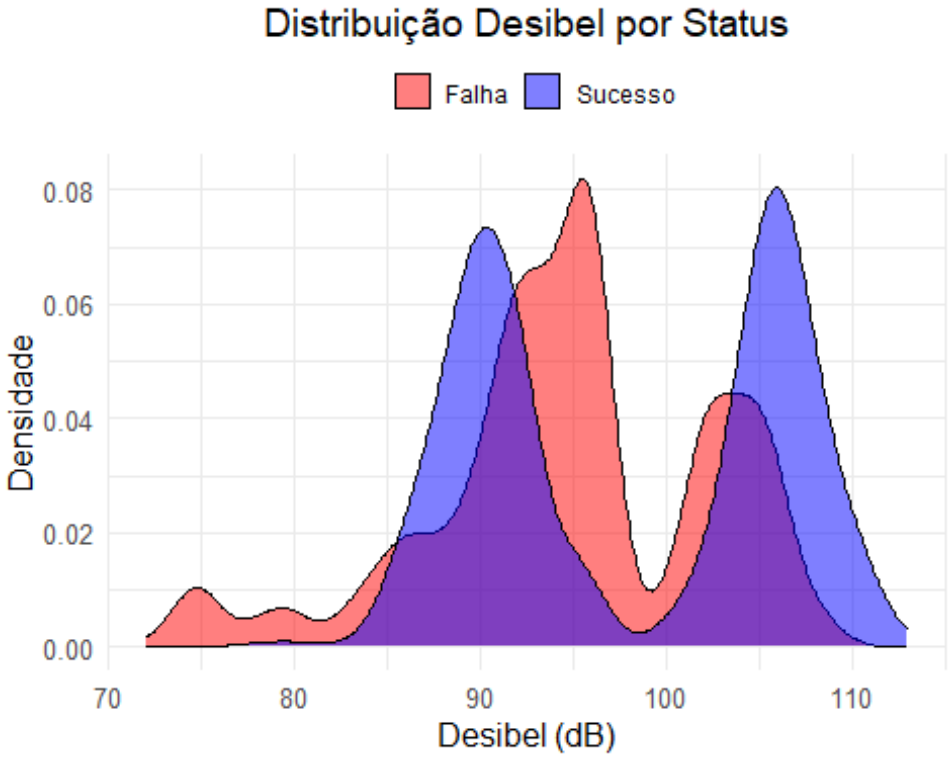
3. Qual a relação entre frequência e resultado do experimento?



Ao analisar o resultado do gráfico de densidade, observa-se que quando a frequência está próxima de 6 Hz, há um maior número de falhas. À medida que a frequência aumenta para 8 Hz, foi observado um aumento no número de sucessos em comparação com as falhas. Esse padrão é observado até 40 Hz. No entanto, a partir de 40 Hz em diante, o número de falhas é superior ao número de sucessos.

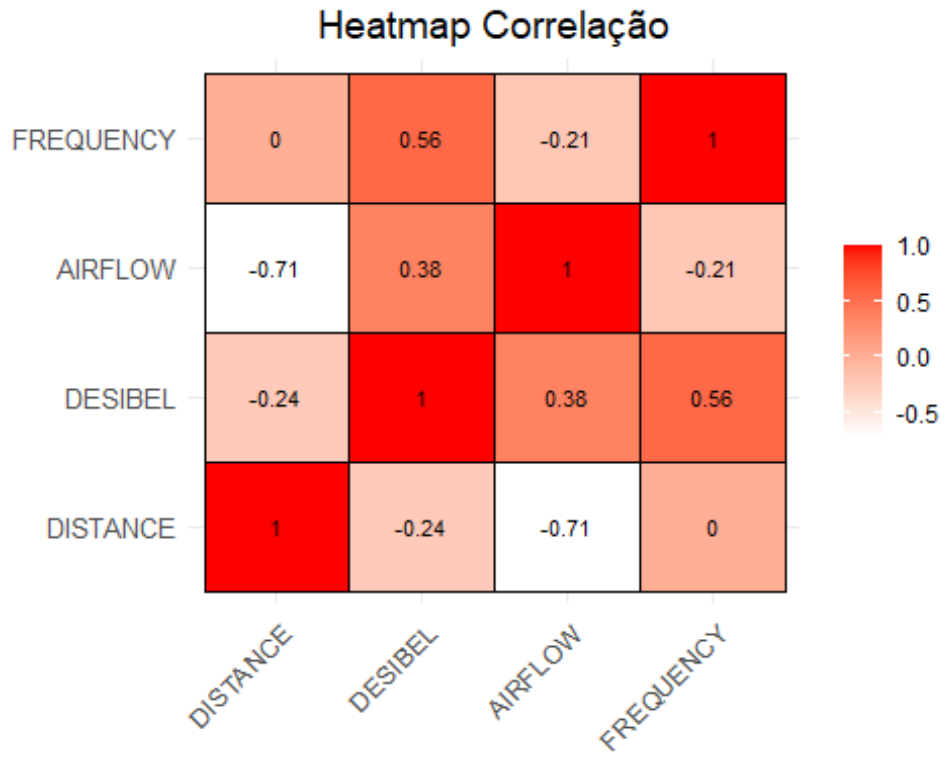
Desibel e Status

4. Como o resultado do experimeto é alterado a medida que a decibel aumenta?



O número de sucessos e falhas no experimento apresentou um comportamento semelhante. Observamos poucos casos onde o número de falhas é superior. Ao nos aproximarmos de 85 dB, podemos observar um aumento nas observações para ambos os casos, seguido de uma queda aos 90 dB para o número de sucesso e para as falhas essa queda se da aos 95 dB. Um padrão um pouco semelhante ocorre quando o experimento atinge os 100 dB.

Correlação



As variáveis Distance e Airflow apresentaram uma alta correlação negativa (-0,71), o que indica uma associação forte e inversa entre elas. Essa correlação próxima de -1 sugere que, à medida que a distância aumenta, o fluxo de ar diminui e vice-versa. Essas variáveis requerem atenção especial durante a criação do modelo, pois a alta correlação entre elas pode causar problemas de multicolinearidade.

A multicolinearidade ocorre quando duas ou mais variáveis independentes estão altamente correlacionadas entre si. Isso pode afetar negativamente a interpretação dos coeficientes estimados e a estabilidade do modelo. Para evitar problemas decorrentes da multicolinearidade, é importante considerar estratégias como a remoção de uma das variáveis altamente correlacionadas.

Pré-Processamento

O pré-processamento de dados é uma etapa essencial na preparação de dados para a análise e construção de modelos de machine learning. Consiste em uma série de técnicas e procedimentos aplicados aos dados brutos com o objetivo de melhorar a qualidade dos dados, reduzir ruídos, tratar valores ausentes e outliers, além de preparar os dados de acordo com as necessidades específicas do modelo.

Label Encoding

O Label Encoding é uma técnica de pré-processamento de dados usada para transformar variáveis categóricas em valores numéricos. Ele atribui um valor inteiro

único para cada categoria presente na variável, permitindo que algoritmos de machine learning processem esses dados.

Aplicando label encoding na variável FUEL

```
df$FUEL <- recode(df$FUEL,
                  "gasoline" = 1,
                  "kerosene" = 2,
                  "thinner" = 3,
                  "lpg" = 4)
```

Padronização

A padronização dos dados é uma técnica de pré-processamento que visa transformar as variáveis em uma escala com média zero e desvio padrão igual a um. Essa técnica é importante para garantir que as variáveis estejam na mesma escala e tenham a mesma ordem de grandeza.

Aplicando padronização

```
df[, num] <- scale(df[, num])
```

Treino e teste

A divisão dos dados em treino e teste é essencial para avaliar a capacidade de generalização de modelos de machine learning. Uma parte dos dados é usada para treinar o modelo e outra parte é reservada para testá-lo. Isso evita que o modelo se ajuste excessivamente aos dados de treinamento.

Aplicando divisão em dados de treino e teste

```
indice_treinamento <- createDataPartition(df$FUEL, p = 0.7, list = FALSE)
dados_treinamento <- df[indice_treinamento,]
dados_teste <- df[-indice_treinamento,]
```

Machine Learning

A aprendizagem de máquina, também conhecida como machine learning, é um campo da inteligência artificial que se concentra no desenvolvimento de algoritmos e técnicas que permitem aos computadores aprenderem e melhorarem automaticamente a partir de dados. Em vez de serem explicitamente programadas para realizar uma tarefa específica, as máquinas aprendem com exemplos e experiências anteriores, permitindo que tomem decisões e realizem previsões com base em padrões e informações presentes nos dados.

GLM - Generalized Linear Model

O modelo Generalized Linear Model (GLM) é uma abordagem estatística para modelagem de dados que permite analisar relações entre variáveis. Ele estende o modelo linear tradicional ao lidar com diferentes tipos de variáveis de resposta, incluindo binárias, de contagem e contínuas.

```r
# Treinando o modelo de regressão logística - GLM
glm_v1 <- train(STATUS ~ .,
                data = dados_treinamento,
                method = "glm",
                family = "binomial")
```

Descrição e resultado do modelo glm_v1:

```r
glm_v1
```

```
## Generalized Linear Model
##
## 12210 samples
##     6 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 12210, 12210, 12210, 12210, 12210, 12210, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8979826  0.7958767
```

- O modelo é composto por 12.210 amostras, com 6 variáveis preditoras, classificadas em duas classes: “0” (falhas) e “1” (sucesso) na extinção da chama.
- Para avaliar o desempenho do modelo, utilizou-se a técnica de reamostragem chamada Bootstrap, que gera múltiplas amostras do mesmo tamanho do conjunto de dados original, permitindo avaliar a estabilidade e variação do desempenho do modelo.
- Observou-se uma acurácia aproximada de 0,90, indicando que o modelo apresenta boa qualidade de classificação. Isso significa que o modelo tem um bom desempenho na tarefa de classificar o objeto de estudo.

Previsão para dados de teste:

```r
# Fazer previsões nos dados de teste
previsoes_glm_v1 <- predict(glm_v1, newdata = dados_teste)
```

observando acurácia para previsão com os dados de teste:

```r
# Calcular a acurácia
acuracia_glm_v1 <- confusionMatrix(previsoes_glm_v1,
dados_teste$STATUS)$overall['Accuracy']

acuracia_glm_v1
```

```
##  Accuracy
## 0.8956422
```

A acurácia obtida ao prever os dados de teste continuou em torno de 0,90. Isso indica que temos um bom modelo capaz de generalizar o padrão dos dados, equilibrando a capacidade de generalização e aprendizado, evitando possíveis problemas de overfitting e underfitting.

```
matriz_confusao_glm_v1 <- confusionMatrix(previsoes_glm_v1,
dados_teste$STATUS)
```

Matriz de confusão:

```
matriz_confusao_glm_v1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2364  333
##          1  213 2322
##
##                Accuracy : 0.8956
##                  95% CI : (0.887, 0.9038)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7914
##
##  Mcnemar's Test P-Value : 3.529e-07
##
##             Sensitivity : 0.9173
##             Specificity : 0.8746
##          Pos Pred Value : 0.8765
##          Neg Pred Value : 0.9160
##              Prevalence : 0.4925
##          Detection Rate : 0.4518
##    Detection Prevalence : 0.5155
##       Balanced Accuracy : 0.8960
##
##        'Positive' Class : 0
##
```

- O modelo previu corretamente a classe 0 em 2364 instâncias, chamado de verdadeiro negativo.
- Houve 213 casos em que o valor esperado era 0, mas o modelo previu 1, o que representa um falso positivo.
- Em cerca de 333 ocasiões, o modelo previu 0 quando o valor esperado era 1, caracterizando um falso negativo.
- O modelo acertou a classe 1 em 2322 instâncias, conhecido como verdadeiro positivo.

KNN - k-nearest neighbors

O KNN (k-nearest neighbors) é um algoritmo de aprendizado de máquina usado para classificação e regressão. Ele classifica um ponto de dados com base na classe da maioria dos k vizinhos mais próximos no espaço de características. O valor de k determina a influência dos vizinhos na classificação. O KNN é simples e intuitivo, porém pode ser computacionalmente caro para conjuntos de dados grandes. Ele não faz suposições sobre a distribuição dos dados, tornando-o um método versátil para problemas diversos.

Arquivo de controle:

```
ctrl <- trainControl(method = "repeatedcv", repeats = 3)
```

Criando modelo KNN:

```
knn_v1 <- train(STATUS ~ .,
                data = dados_treinamento,
                method = "knn",
                trControl = ctrl,
                tuneLength = 20)
```

Descrição e resultado do modelo knn_v1:

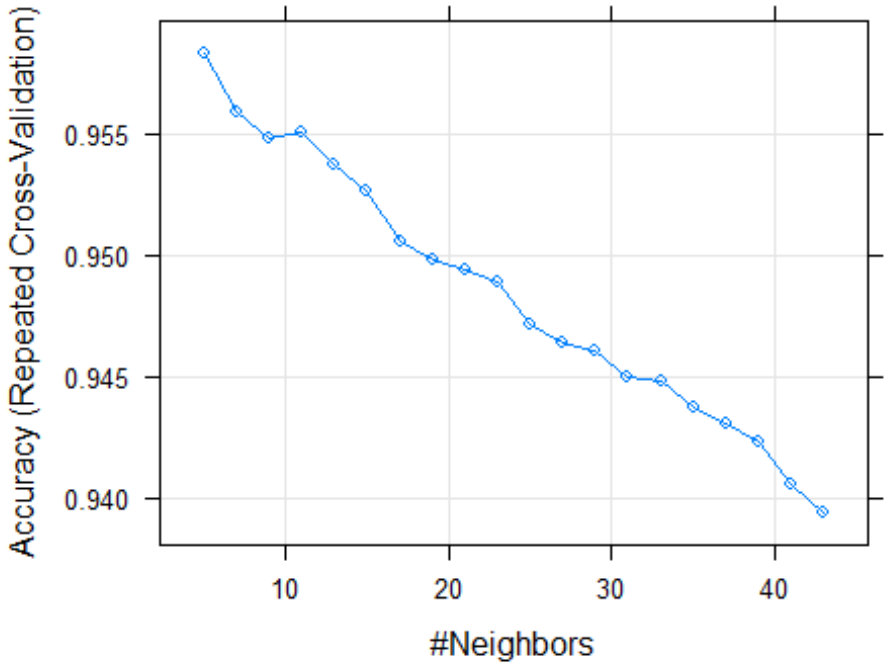
```
knn_v1
```

```
## k-Nearest Neighbors
##
## 12210 samples
##     6 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 10989, 10990, 10988, 10989, 10988, 10989, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.9582854  0.9165463
##    7  0.9558825  0.9117407
##    9  0.9547634  0.9095017
##   11  0.9550091  0.9099943
##   13  0.9536717  0.9073176
##   15  0.9526616  0.9052995
##   17  0.9505595  0.9010960
##   19  0.9497951  0.8995679
##   21  0.9493581  0.8986954
##   23  0.9488941  0.8977688
##   25  0.9471742  0.8943285
##   27  0.9463824  0.8927475
##   29  0.9460822  0.8921485
```

```
##   31  0.9449357  0.8898559
##   33  0.9448267  0.8896401
##   35  0.9437070  0.8874035
##   37  0.9430791  0.8861480
##   39  0.9422876  0.8845656
##   41  0.9405952  0.8811822
##   43  0.9393939  0.8787775
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

- O modelo utilizou uma amostra de 12.210 instâncias e 6 variáveis preditoras. A variável de resposta possui duas classes: “0” (falhas) e “1” (sucesso) na extinção da chama.
- Foram testados diferentes valores de k (número de vizinhos mais próximos considerados) e suas respectivas métricas de desempenho foram registradas. O valor de k = 5 apresentou a maior acurácia, aproximadamente 0.96. Isso indica que o modelo possui um bom desempenho na tarefa de classificação do objeto de estudo.

plot acurácia e visinhos proximos:



- Ao observar o plot do modelo **knn_v1**, nota-se que conforme o número de k (número de vizinhos mais próximos considerados) aumenta, a acurácia diminui.

Previsão para dados de teste:

```
knnpredict <- predict(knn_v1, newdata = dados_teste)
```

observando acurácia para previsão com os dados de teste:

```
knnacuracia <- confusionMatrix(knnpredict,
dados_teste$STATUS)$overall['Accuracy']

knnacuracia

##  Accuracy
## 0.9581422
```

A acurácia obtida no modelo knn ao fazer uma previsão com os dados de teste foi de aproximadamente 0.96. Indicativo de que temos um bom modelo, assim como o anterior, capaz de generalizar o padrão dos dados, equilibrando a capacidade de generalização e aprendizado.

```
knnmatriz_confusao <- confusionMatrix(knnpredict, dados_teste$STATUS)
```

Matriz de confusão:

```
knnmatriz_confusao

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2494  136
##          1   83 2519
##
##                Accuracy : 0.9581
##                  95% CI : (0.9524, 0.9634)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9163
##
##  Mcnemar's Test P-Value : 0.0004417
##
##             Sensitivity : 0.9678
##             Specificity : 0.9488
##          Pos Pred Value : 0.9483
##          Neg Pred Value : 0.9681
##              Prevalence : 0.4925
##          Detection Rate : 0.4767
##    Detection Prevalence : 0.5027
##       Balanced Accuracy : 0.9583
##
##        'Positive' Class : 0
##
```

- O modelo previu corretamente a classe 0 em 2495 instâncias, chamado de verdadeiro negativo.
- Houve 82 casos em que o valor esperado era 0, mas o modelo previu 1, o que representa um falso positivo.
- Em cerca de 136 ocasiões, o modelo previu 0 quando o valor esperado era 1, caracterizando um falso negativo.
- O modelo acertou a classe 1 em 2519 instâncias, conhecido como verdadeiro positivo.

O modelo KNN apresentou uma melhora significativa em comparação com o modelo GLM. Houve um aumento na acurácia e uma redução tanto nos falsos positivos quanto nos falsos negativos.

NB - Naive Bayes

O modelo Naive Bayes é um algoritmo de aprendizado de máquina baseado no teorema de Bayes. Ele assume uma independência condicional entre os recursos, o que significa que cada recurso é tratado independentemente dos outros. Esse modelo é amplamente utilizado em tarefas de classificação e é conhecido por sua eficiência computacional e facilidade de implementação. O Naive Bayes calcula a probabilidade condicional de uma classe dado um conjunto de recursos usando a fórmula de Bayes.

Criando modelo NB:

```
modelo_nb <- naiveBayes(STATUS ~ .,
                        data = dados_treinamento)
```

Previsão para dados de teste:

```
predict_nb <- predict(modelo_nb, newdata = dados_teste)
```

observando acurácia para previsão com os dados de teste:

```
acuracia_nb <- confusionMatrix(predict_nb,
dados_teste$STATUS)$overall['Accuracy']

acuracia_nb

##  Accuracy
## 0.8738532
```

A acurácia obtida ao prever os dados de teste apresentou um valor de aproximadamente 0,87. Embora isso indique que temos um bom modelo, esse valor é inferior ao resultado obtido no modelo anterior, o KNN.

```
matriz_confusaonb <- confusionMatrix(predict_nb, dados_teste$STATUS)
```

Matriz de confusão:

```
matriz_confusaonb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2370  453
##          1  207 2202
##
##                Accuracy : 0.8739
##                  95% CI : (0.8646, 0.8827)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.748
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9197
##             Specificity : 0.8294
##          Pos Pred Value : 0.8395
##          Neg Pred Value : 0.9141
##              Prevalence : 0.4925
##          Detection Rate : 0.4530
##    Detection Prevalence : 0.5396
##       Balanced Accuracy : 0.8745
##
##        'Positive' Class : 0
##
```

- O modelo previu corretamente a classe 0 em 2370 instâncias, chamado de verdadeiro negativo.
- Houve 207 casos em que o valor esperado era 0, mas o modelo previu 1, o que representa um falso positivo.
- Em cerca de 453 ocasiões, o modelo previu 0 quando o valor esperado era 1, caracterizando um falso negativo.
- O modelo acertou a classe 1 em 2202 instâncias, conhecido como verdadeiro positivo.

RF - Random Forest

A Random Forest é um algoritmo de aprendizado de máquina que combina várias árvores de decisão para realizar tarefas de classificação ou regressão. Cada árvore na floresta é treinada em uma amostra aleatória dos dados de treinamento, e a previsão final é obtida por votação majoritária (classificação) ou média (regressão) das previsões de todas as árvores. Esse modelo é conhecido por sua capacidade de lidar com grandes conjuntos de dados e por sua robustez contra overfitting. Além

disso, a Random Forest pode fornecer importância relativa dos recursos, ajudando na seleção de características.

Criando modelo RF:

```
modelo_rf <- randomForest(STATUS ~ .,
                          data = dados_treinamento,
                          ntree = 500,
                          mtry = 5)
```

Descrição e resultado do modelo modelo_rf:

```
modelo_rf
```

```
##
## Call:
##  randomForest(formula = STATUS ~ ., data = dados_treinamento,
ntree = 500, mtry = 5)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 2.92%
## Confusion matrix:
##      0    1 class.error
## 0 6009  173  0.02798447
## 1  184 5844  0.03052422
```

- o modelo de Random Forest foi aplicado a um problema de classificação. Ele foi treinado usando os dados de treinamento e consiste em 500 árvores de decisão. Durante a construção das árvores, 5 variáveis foram consideradas em cada divisão.
- O modelo estimou que a taxa de erro fora da amostra (OOB) é de aproximadamente 2,98%. Isso significa que, em média, o modelo tem uma taxa de acerto de cerca de 97,02%

Previsão para dados de teste:

```
predict_rf <- predict(modelo_rf, newdata = dados_teste)
```

observando acurácia para previsão com os dados de teste:

```
acuracia_rf <- confusionMatrix(predict_rf,
dados_teste$STATUS)$overall['Accuracy']

acuracia_rf
```

```
##  Accuracy
## 0.9701835
```

A acurácia obtida ao prever os dados de teste apresentou um valor de aproximadamente 0,97. Isso indica que temos um bom modelo, e ele é capaz de generalizar o padrão dos dados, equilibrando a capacidade de generalização e aprendizado, evitando possíveis problemas de overfitting e underfitting.

```
matriz_confusaorf <- confusionMatrix(predict_rf, dados_teste$STATUS)
```

Matriz de confusão:

```
matriz_confusaorf
```

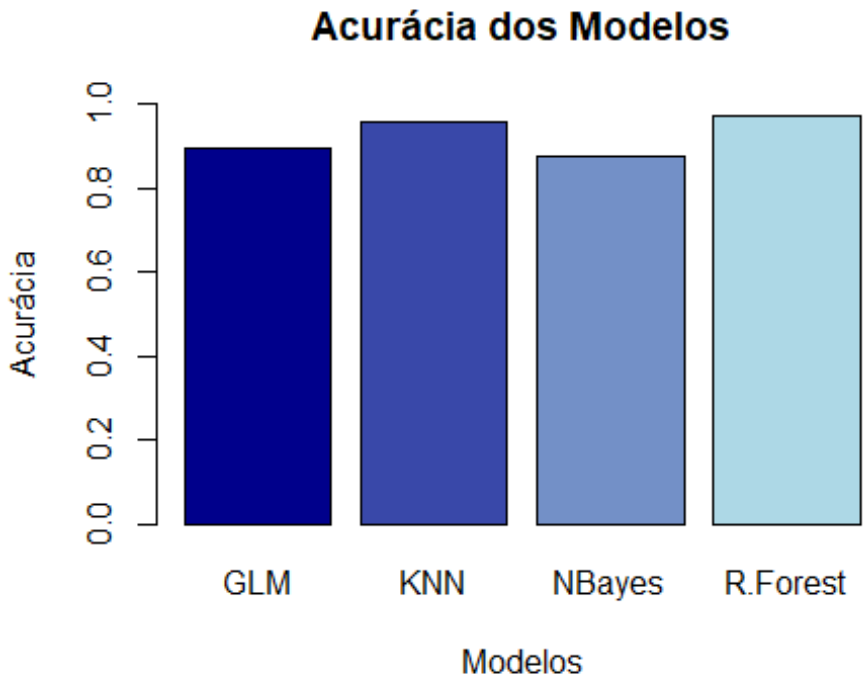
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2515   94
##          1   62 2561
##
##                Accuracy : 0.9702
##                  95% CI : (0.9652, 0.9746)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9404
##
##  Mcnemar's Test P-Value : 0.01307
##
##             Sensitivity : 0.9759
##             Specificity : 0.9646
##          Pos Pred Value : 0.9640
##          Neg Pred Value : 0.9764
##              Prevalence : 0.4925
##          Detection Rate : 0.4807
##    Detection Prevalence : 0.4987
##       Balanced Accuracy : 0.9703
##
##        'Positive' Class : 0
##
```

- O modelo previu corretamente a classe 0 em 2513 instâncias, chamado de verdadeiro negativo.
- Houve 64 casos em que o valor esperado era 0, mas o modelo previu 1, o que representa um falso positivo.
- Em cerca de 93 ocasiões, o modelo previu 0 quando o valor esperado era 1, caracterizando um falso negativo.
- O modelo acertou a classe 1 em 2562 instâncias, conhecido como verdadeiro positivo.

O modelo Random Forest apresentou o melhor desempenho se comparado aonde mais modelos. Sua precisão de prvisão aumentou, reduzindo os erros dos falsos positivos e falsos nagativos ao fazer uma previsão com os dados de teste.

Conclusão

Observando os dados de acurácia do gráfico seguinte, embora todos os modelos tenham apresentado um bom desempenho, o modelo Random Forest foi o que obteve a maior acurácia.



Através desta pesquisa, utilizando simulações feitas no computador e a criação de modelos de Machine Learning, foi possível adicionar uma camada adicional de segurança ao processo de teste de extintores de incêndio. Os dados utilizados e os modelos desenvolvidos demonstraram uma capacidade elevada de prever a eficácia dos extintores em extinguir as chamas. Isso representa um avanço significativo na área de segurança, oferecendo uma abordagem mais precisa e confiável na avaliação do desempenho dos extintores. Esses modelos podem contribuir para aprimorar os procedimentos de teste e garantir uma resposta mais eficaz em situações de incêndio, aumentando a segurança das pessoas e dos ambientes.