# Desafio Senior Labs Challenge

O objetivo desse documento é apresentar soluções para o desafio da Senior Labs Challenge.
Nesse desafio foi proposto criar um artigo detalhando qual a metodologia aplicada e desenvolver soluções através de ferramentas de software para extrair dados estatísticos e relatórios gráficos da base de dados disponibilizada.

## 1. Introdução

Nas últimas décadas, o e-mail se tornou uma das principais formas comunicação no meio coorporativo, é cada vez mais comum que as pessoas sejam inundadas por uma grande quantidade de mensagens.
Infelizmente, muitas dessas mensagens são spams, são enviadas com o objetivo de promover produtos ou serviços de maneira invasiva e não solicitada.

Além de ser desagradável o recebimento dessas mensagens, os spams também podem representar uma ameaça à segurança dos usuários, já que muitos desses e-mails podem conter links maliciosos que podem infectar o computador com vírus ou roubar informações pessoais.

No desafio proposto, foi disponibilizado uma base de dados no formato CSV contém vários exemplos de mensagens comuns e de spams.

Utilizando essas informações, o objetivo é extrair dados estatísticos, relatório gráficos e por fim propor um modelo que seja possível classificar de forma automática se o e-mail é uma mensagem comum ou spam.

## 2. Problema

Extrair informações estáticas da base de dados disponibilizada, através de qualquer software ou linguagem de programação.

## 3. Objetivo

Realizar a criação de um artigo evidenciando os dados extraídos, consolidados em relatórios e apresentar metodologia aplicada para obter os resultados esperados.

## 4. Solução Proposta

O software utilizado para extrair as informações solicitadas é o Power BI da Microsoft,
Através dele foi possível importar a base de dados e gerar os relatórios solicitados por meio dos recursos e tecnologia da ferramenta.

## 6. Primeira Etapa

Primeira etapa do desafio consiste em extrair estatísticas desta base de dados

### 6.1 Exibir gráfico as palavras

Exibir gráfico as palavras mais frequentes em toda a base de dados (Ex.: gráfico de barras, nuvem de palavras, etc).

#### 6.1.1 Solução implementada

Através de um modelo gráfico “Nuvem de Palavras”, foi gerado uma representação visual da frequência de palavras em um texto ou conjunto de dados.

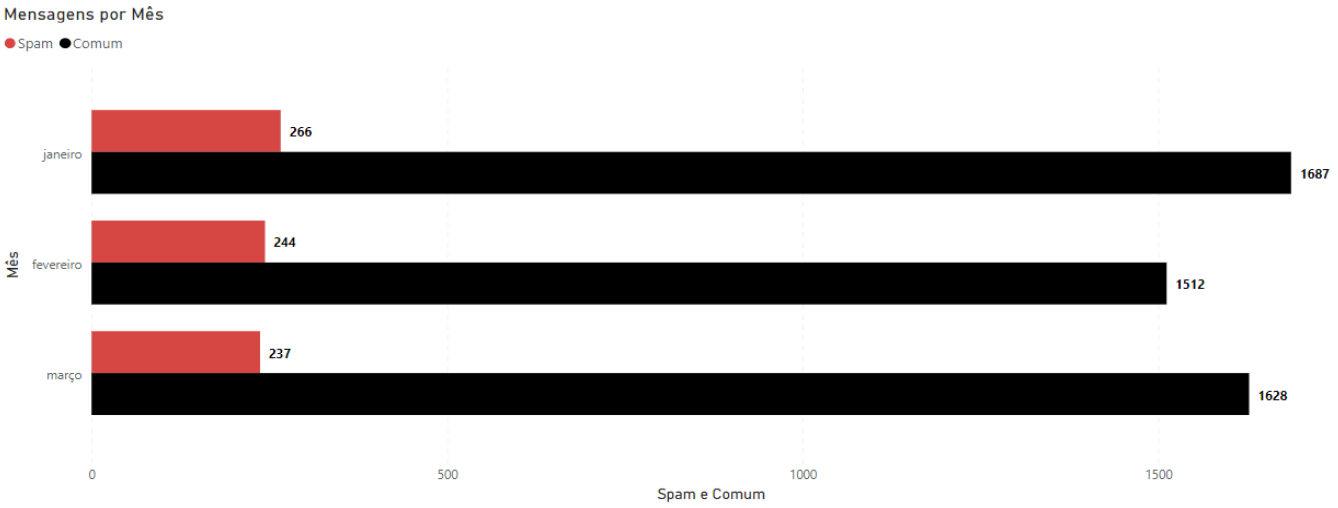
As palavras extraídas desse gráfico são referentes ao campo “Full_Text”, sem filtros, os dados foram exportados considerando todas as mensagens de e-mails.

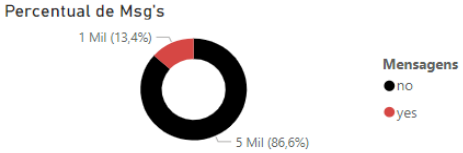

### 6.2 Quantidades de mensagens comuns e spams

Exibir gráfico com as quantidades de mensagens comuns e spams para cada mês.

### 6.2 Solução implementada

Através de um modelo gráfico “Gráfico de barras”, é exibido a quantidade de mensagens comuns e spam por mês, uma matriz com os totalizadores e um “Gráfico em Pizza” com os respectivos percentuais.




| Mês | Spam | Comum | Total |
|---|---|---|---|
| janeiro | 266 | 1687 | 1953 |
| fevereiro | 244 | 1512 | 1756 |
| março | 237 | 1628 | 1865 |
| **Total** | **747** | **4827** | **5574** |

## 6.3 Calcular indicadores do campo “Word_Count” por Mês

Calcular o máximo, o mínimo, a média, a mediana, o desvio padrão e a variância da quantidade total de palavras (Word_Count) para cada mês;

### 6.3.1 Solução implementada

Abaixo os resultados para cada grupo de informações, a consolidação dos dados foi executada através do agrupamento dos dados por mês além das respectivas fórmulas para cálculo.

| Mês | Máximo de Palavras |
|---|---|
| janeiro | 190 |
| março | 115 |
| fevereiro | 100 |
| **Total** | **190** |

| Mês | Média de Palavras |
|---|---|
| janeiro | 16,34 |
| fevereiro | 16,03 |
| março | 16,29 |
| **Total** | **16,22** |

| Mês | Desvio padrão de Palavras |
|---|---|
| janeiro | 12,55 |
| fevereiro | 11,04 |
| março | 11,57 |
| **Total** | **11,77** |

| Mês | Mínimo de Palavras |
|---|---|
| janeiro | 2 |
| fevereiro | 2 |
| março | 2 |
| **Total** | **2** |

| Mês | Mediana de Palavras |
|---|---|
| janeiro | 13 |
| fevereiro | 13 |
| março | 12 |
| **Total** | **13** |

| Mês | Variação de Palavras |
|---|---|
| janeiro | 157,60 |
| fevereiro | 121,87 |
| março | 133,94 |
| **Total** | **138,44** |

## 6.4 Sequência de mensagens comuns por Mês

Exibir o dia de cada mês que possui a maior sequência de mensagens comuns (não spam).

### 6.4.1 Solução implementada

Dashboard exibindo informações dos dias de cada mês com maior sequência de mensagens comuns envidas. Para chegar nesse resultado foi necessário criar um novo modelo de dados agrupando as informações por mês e filtrando apenas e-mails que não são spams.

| Mês | Emails por dia |
|---|---|
| ⊟ **janeiro** | **69** |
| domingo, 1 de janeiro de 2017 | 69 |
| ⊟ **março** | **69** |
| quarta-feira, 8 de março de 2017 | 69 |
| ⊟ **fevereiro** | **72** |
| segunda-feira, 13 de fevereiro de 2017 | 72 |

## 7.0 Segunda etapa

A segunda etapa consiste em aplicar um método capaz de classificar automaticamente as mensagens como “comum” e “spam”. Como você considera os resultados encontrados? Justifique.

## 7.1 Descrição da Solução

A solução de machine learning apresentada é um modelo de classificação que tem como objetivo identificar se um e-mail é considerado spam ou não. O modelo é treinado utilizando um conjunto de dados que possui informações sobre os e-mails, incluindo a presença de palavras específicas, o tamanho do e-mail em termos de número de palavras, a frequência de ocorrência de determinadas palavras, entre outras.

O conjunto de dados é pré-processado para transformar as informações em um formato adequado para a alimentação do modelo de aprendizado de máquina. Em seguida, um modelo de classificação é escolhido e treinado usando os dados de treinamento.

Após o treinamento, o modelo é avaliado usando um conjunto de dados de teste separado. A precisão do modelo é calculada a partir das previsões feitas pelo modelo para o conjunto de teste. Uma vez que o modelo foi avaliado e ajustado para obter o melhor desempenho possível, ele está pronto para ser utilizado em novos dados.

No exemplo apresentado, foram utilizados vários algoritmos de classificação, como Regressão Logística, Árvores de Decisão, e Naive Bayes. Também foram utilizadas técnicas de pré-processamento de dados, como vetorização de texto, que é uma técnica comum para processar dados de texto para alimentação de modelos de aprendizado de máquina.

### 7.2 Solução Implementada

```
import pandas as pd

#Url contendo os dados que serão utilizamos para criação do modelo
urlDbGit = "https://raw.githubusercontent.com/brunolimawk/LabsChallenge/main/DataBase.csv"

#Busca os dados da tabela no formato CSV e cria um DataFrame
df = pd.read_csv(urlDbGit, delimiter=";")

#Criar colunas contendo o mínimo, a média, o máximo, o desvio padrão,
mediana e a variação de palavras da coluna Full_Text

df['Min_palavras'] = df['Full_Text'].str.split().apply(lambda x: len(x)).min()
df['Media_palavras'] = df['Full_Text'].str.split().apply(lambda x: len(x)).mean()
df['Max_palavras'] = df['Full_Text'].str.split().apply(lambda x: len(x)).max()
df['Desvio_palavras'] = df['Full_Text'].str.split().apply(lambda x: len(x)).std()
df['Mediana_palavras'] = df['Full_Text'].str.split().apply(lambda x: len(x)).median()
#df['Variacao_palavras'] = df['Full_Text'].str.split().apply(lambda x: max(x) - min(x))

#Definir a coluna "IsSpam" como variável alvo e transformá-la em binarios
df['IsSpam'] = df['IsSpam'].apply(lambda x: 1 if x == 'Yes' else 0)


#Separar o conjunto de dados em treinamento e teste
df['IsSpam'] = df['IsSpam'].apply(lambda x: 1 if x == 'Yes' else 0)

#Divide o modelo e prepara o treinamento
from sklearn.model_selection import train_test_split

treinamento, teste = train_test_split(df, test_size=0.2, random_state=42)


#Importar o CountVectorizer e utiliza-
lo para transformar a coluna "Full_Text" em uma matriz
from sklearn.feature_extraction.text import CountVectorizer

count_vectorizer = CountVectorizer()
X_treinamento = count_vectorizer.fit_transform(treinamento['Full_Text'])
X_teste = count_vectorizer.transform(teste['Full_Text'])
y_treinamento = treinamento['IsSpam']
y_teste = teste['IsSpam']


#Adicionar dados de treinamento e teste
import numpy as np

X_treinamento = np.hstack((X_treinamento.toarray(), treinamento[['Min_palavras', 'Media_pa
lavras', 'Max_palavras', 'Desvio_palavras', 'Mediana_palavras']].values))
X_teste = np.hstack((X_teste.toarray(), teste[['Min_palavras', 'Media_palavras', 'Max_pala
vras', 'Desvio_palavras', 'Mediana_palavras']].values))
```

```
#Treinar um modelo de classificação, utilizando Naive Bayes
from sklearn.naive_bayes import MultinomialNB

modelo = MultinomialNB()
modelo.fit(X_treinamento, y_treinamento)

#Avaliar o desempenho do modelo
from sklearn.metrics import accuracy_score

#Fazer as previsões com o modelo treinado
y_pred = modelo.predict(X_teste)


# Calcular acurácia
acuracia = accuracy_score(y_teste, y_pred)

 print("Acurácia:", acuracia)
```

## 7.2 Conclusão

A união das métricas apresentadas aplicadas a um modelo de dados treinado, é possível classificar qual o tipo de mensagem que está sendo transmitida, com pequena margem de desvio padrão.

O modelo apresentando é uma amostragem com poucos dados analisados, para que se tenha uma assertividade maior, é importante que seja aplicado em um base com maior numero de registros, dessa forma o modelo terá um referencial mais apurado, aumentando as possibilidades de acerto.

## 8 Referencias


Documentação Power BI: https://learn.microsoft.com/pt-br/power-bi/consumer/

Instruções desafio: https://github.com/SeniorSA/seniorlabs-challenge

Funções Python: https://pandas.pydata.org/

Esclarecer dúvidas: https://stackoverflow.com/