

# Drinking water Potabilidade

Primeiramente importaremos e mostraremos como se comporta a base.

Instruções Instalação de Módulos Instale se necessário e comente novamente cada linha abaixo. Se faltar mais alguma biblioteca no seu notebook considere rodar pip install

```
In [82]: #pip install missingno
```

```
In [83]: #pip install plotly
```

```
In [84]: #pip install confusion_matrix
```

```
In [85]: #pip install mlxtend
```

## Bibliotecas

In [86]: # Basic Libraries

```
import numpy as np
import pandas as pd
from warnings import filterwarnings
from collections import Counter

# Visualizations Libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly
import plotly.offline as pyo
import plotly.express as px
import plotly.graph_objs as go
pyo.init_notebook_mode()
import plotly.figure_factory as ff
import missingno as msno

# Data Pre-processing Libraries
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.model_selection import train_test_split

# Modelling Libraries
from sklearn.linear_model import LogisticRegression,RidgeClassifier,SGDClassifier,PassiveAggressiveClassifier
from sklearn.linear_model import Perceptron
from sklearn.svm import SVC,LinearSVC,NuSVC
from sklearn.neighbors import KNeighborsClassifier,NearestCentroid
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB,BernoulliNB
from sklearn.ensemble import VotingClassifier

# Evaluation & CV Libraries
from sklearn.metrics import precision_score,accuracy_score
from sklearn.model_selection import RandomizedSearchCV,GridSearchCV,RepeatedStratifiedKFold
```

## Descrição

1. Sólidos: Total de sólidos dissolvidos em ppm.
2. Cloraminas: Quantidade de cloraminas in ppm.
3. Sulfato: Quantidade de sulfato dissolvido em mg/L.
4. Conductividade: Conductividade elétrica da água em  $\mu\text{S}/\text{cm}$ .
5. Carbono Orgânico: Quantidade de carbono orgânico en ppm.
6. Trihalometanos: Quantidade de trihalometanos en  $\mu\text{g}/\text{L}$ .
7. Turbidez: Medida da dificuldade de um feixe de luz atravessar uma certa quantidade de água em NTU.
8. Potabilidade: Indica se a água é própria para consumo e ingestão, sem risco à saúde. Potável- 1 e não potável - 0

### Unidades de Medida

- ppm - partes por milhão.
- mg/L - miligrama por litro.
- $\mu\text{S}/\text{cm}$  - micro-Siemens por centímetro.
- $\mu\text{g}/\text{L}$  - micrograma por litro.
- NTU - unidade nefelométrica de turbidez.

Observação ao grupo: talvez possam colocar imagens de algumas propriedades acima.

```
In [87]: import pandas as pd
df = pd.read_csv("water_potability.csv")
df.head()
```

Out[87]:

	ph	Dureza	Solidos	Cloraminas	Sulfato	Conductividade	Carbono_Organico	Trihalometanos	Turbidez	Potabilidade
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

```
In [88]: colors_blue = ["#132C33", "#264D58", '#17869E', '#51C4D3', '#B4DBE9']
colors_dark = ['#1F1F1F', "#313131", '#636363', '#AEAEAE', '#DADADA']
colors_green = ['#01411C', '#4B6F44', '#4F7942', '#74C365', '#D0F0C0']
sns.palplot(colors_blue)
sns.palplot(colors_green)
sns.palplot(colors_dark)
```



```
In [89]: d= pd.DataFrame(df['Potabilidade'].value_counts())
fig = px.pie(d,values='Potabilidade',names=['Não potável','Potável'],hole=0.4,opacity=0.6,
             color_discrete_sequence=[colors_green[3],colors_blue[3]],
             labels={'label':'Potabilidade','Potabilidade':'No. de amostras'})

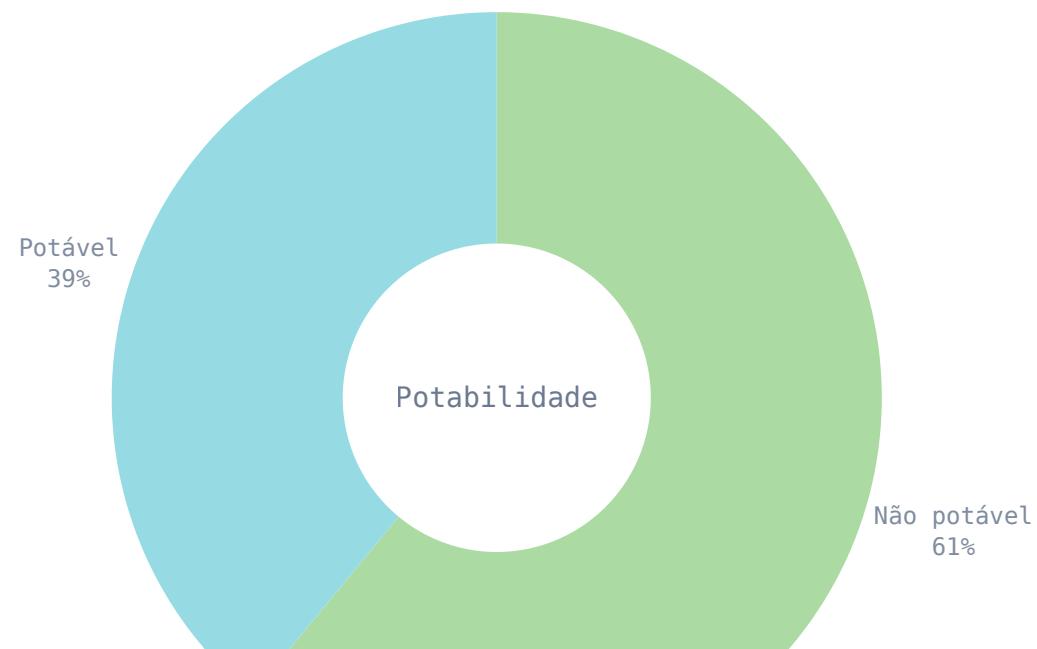
fig.add_annotation(text='Potabilidade',
                    x=0.5,y=0.5,showarrow=False,font_size=14,opacity=0.7,font_family='monospace')

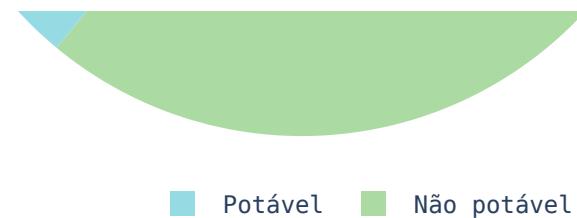
fig.update_layout(
    font_family='monospace',
    title=dict(text='Quantas amostras de água são Potáveis?',x=0.47,y=0.98,
               font=dict(color=colors_dark[2],size=20)),
    legend=dict(x=0.37,y=-0.05,orientation='h',traceorder='reversed'),
    hoverlabel=dict(bgcolor='white'))

fig.update_traces(textposition='outside', textinfo='percent+label')

fig.show()
```

Quantas amostras de água são Potáveis?





## Dureza da água:

Dureza da água é a propriedade relacionada com a concentração de íons de determinados minerais dissolvidos nesta substância ou mais especificamente as águas duras contém sais de cálcio e de magnésio em concentrações relativamente elevadas. No Brasil, a portaria Min. da Saúde N.º 2.914 de 14 de dezembro de 2011 (Anexo X), estabelece o VMP (Valor Máximo Permitido) de 500mg/L de concentração total de Cálcio e Magnésio para que a água seja admitida como potável. Este parâmetro remete a um padrão organoléptico (gosto) da água, já que o sabor eventualmente pode ser considerado uma característica desagradável de águas muito duras.

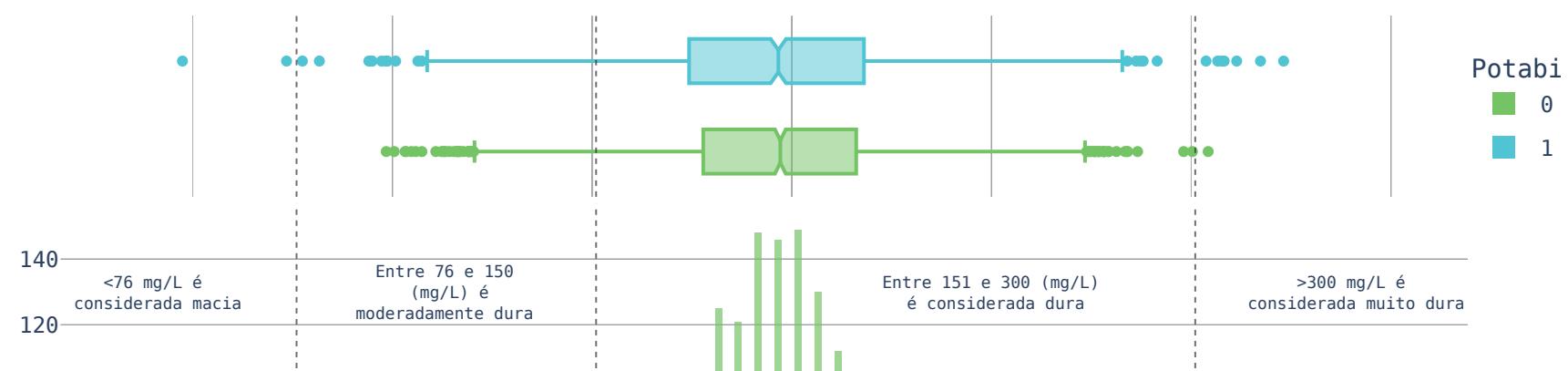
```
In [90]: fig = px.histogram(df,x='Dureza',y=Counter(df['Dureza']),color='Potabilidade',template='plotly_white',
                         marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1],
                         barmode='group',histfunc='count')

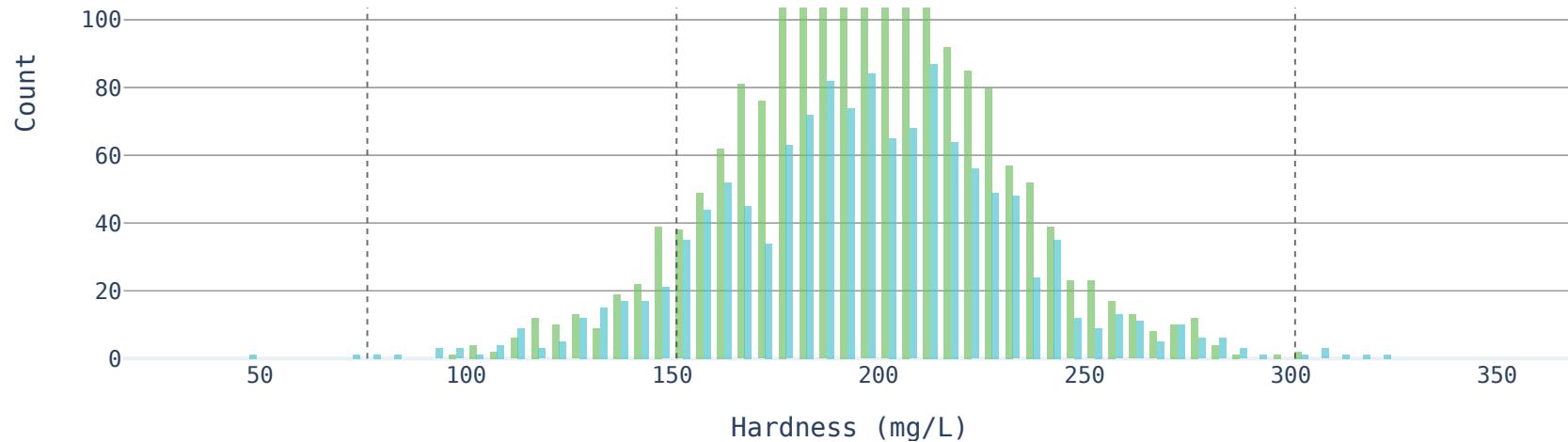
fig.add_vline(x=151, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)
fig.add_vline(x=301, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)
fig.add_vline(x=76, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)

fig.add_annotation(text='<76 mg/L é  
considerada macia',x=40,y=130,showarrow=False,font_size=9)
fig.add_annotation(text='Entre 76 e 150  
(mg/L) é  
moderadamente dura',x=113,y=130,showarrow=False,font_size=9)
fig.add_annotation(text='Entre 151 e 300 (mg/L)  
é considerada dura',x=250,y=130,showarrow=False,font_size=9)
fig.add_annotation(text='>300 mg/L é  
considerada muito dura',x=340,y=130,showarrow=False,font_size=9)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição da concentração de íons minerais dissolvidos',x=0.53,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Hardness (mg/L)',
    yaxis_title_text='Count',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```

Distribuição da concentração de íons minerais dissolvidos





## Nível de pH:

A escala do pH da água pode variar de 1 a 14, indicando a concentração de íons H<sup>+</sup> presentes na água. É essa concentração de íons H<sup>+</sup> que determina o caráter ácido da água. Todo o pH inferior a 6 é ácido. Quanto menor o número, mais ácida é a água. controlado pelo equilíbrio do sistema carbono dióxido-bicarbonato-carbonato.

Um aumento do dióxido de carbono irá abaixar a concentração do nível de pH, enquanto que sua diminuição irá aumentar o pH. A temperatura irá afetar o equilíbrio do pH também. Para uma água pura (não possui nenhuma substância em sua composição) a diminuição do pH cerca de 0.45 ocorre quando a temperatura é aumentada em 25 °C.

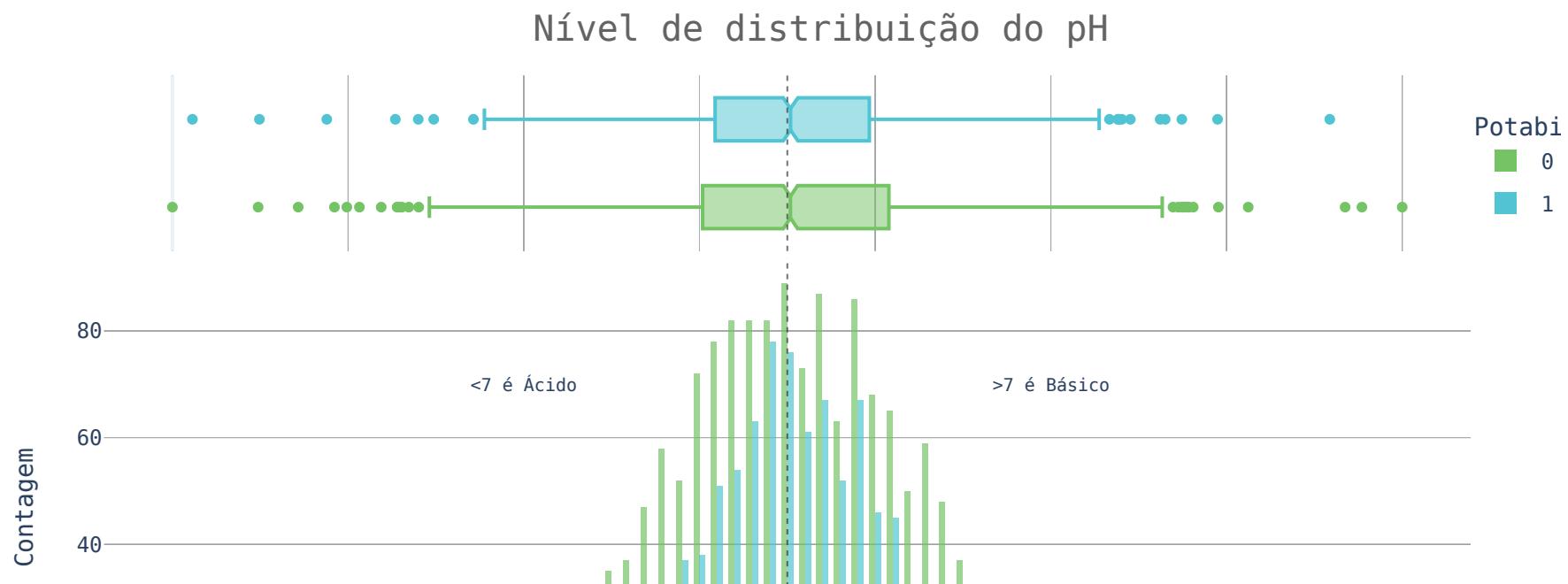
A água potável deve ter um nível de pH entre 6.5–8.5.

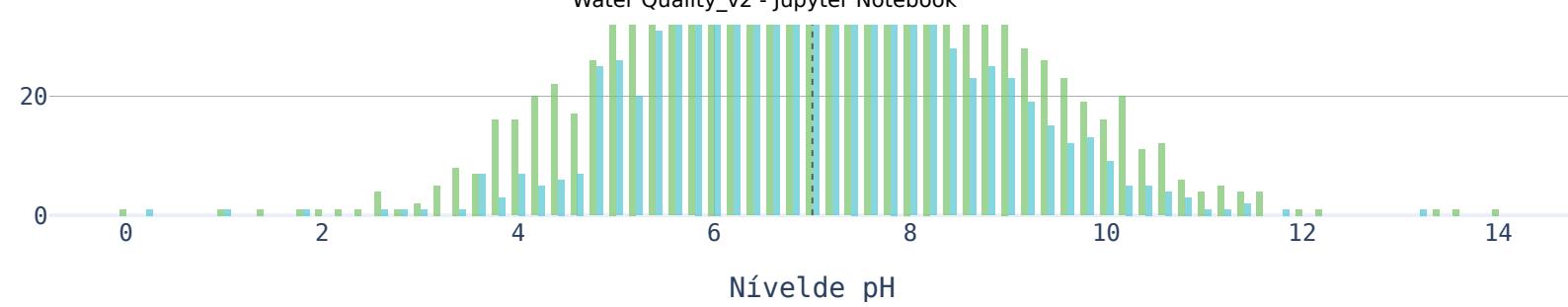
```
In [91]: fig = px.histogram(df,x='ph',y=Counter(df['ph']),color='Potabilidade',template='plotly_white',
                         marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[3]],
                         barmode='group',histfunc='count')

fig.add_vline(x=7, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)

fig.add_annotation(text='<7 é Ácido',x=4,y=70,showarrow=False,font_size=10)
fig.add_annotation(text='>7 é Básico',x=10,y=70,showarrow=False,font_size=10)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Nível de distribuição do pH',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Nível de pH',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```



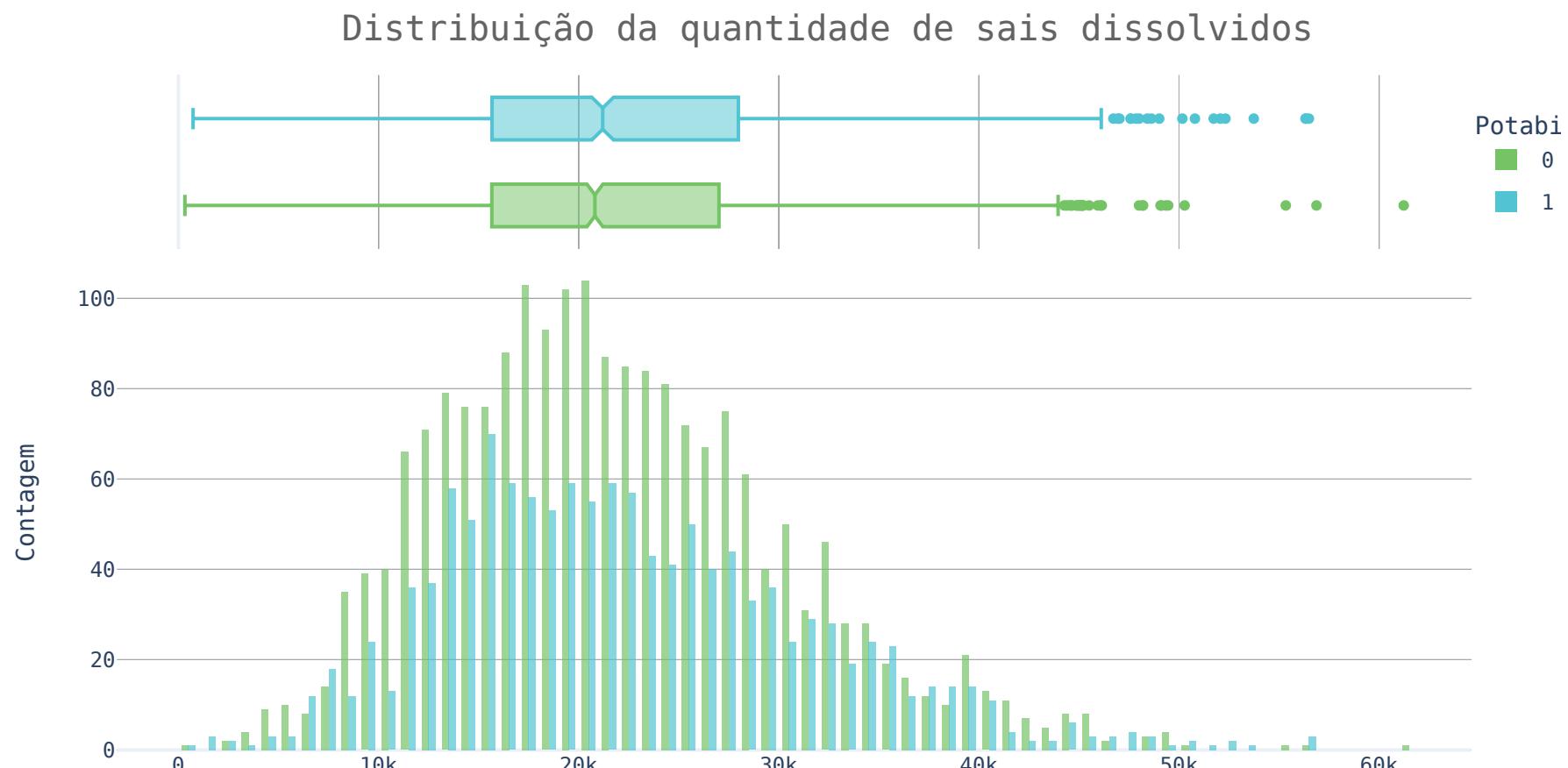


### Níveis de Sólidos Totais Dissolvidos - TDS:

Concentração de sólidos totais dissolvidos (TDS) é medida pela quantidade de materiais dissolvidos na água. O parâmetro de TDS inclui solutos como sódio, cálcio, magnésio, bicarbonato e cloreto que permanecem como um resíduo sólido após a evaporação da água da amostra

```
In [92]: fig = px.histogram(df,x='Solidos',y=Counter(df['Solidos']),color='Potabilidade',template='plotly_white',
                         marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1]],
                         barmode='group',histfunc='count')

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição da quantidade de sais dissolvidos',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Sólidos dissolvidos (ppm)',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```



Sólidos dissolvidos (ppm)

### Cloramina:

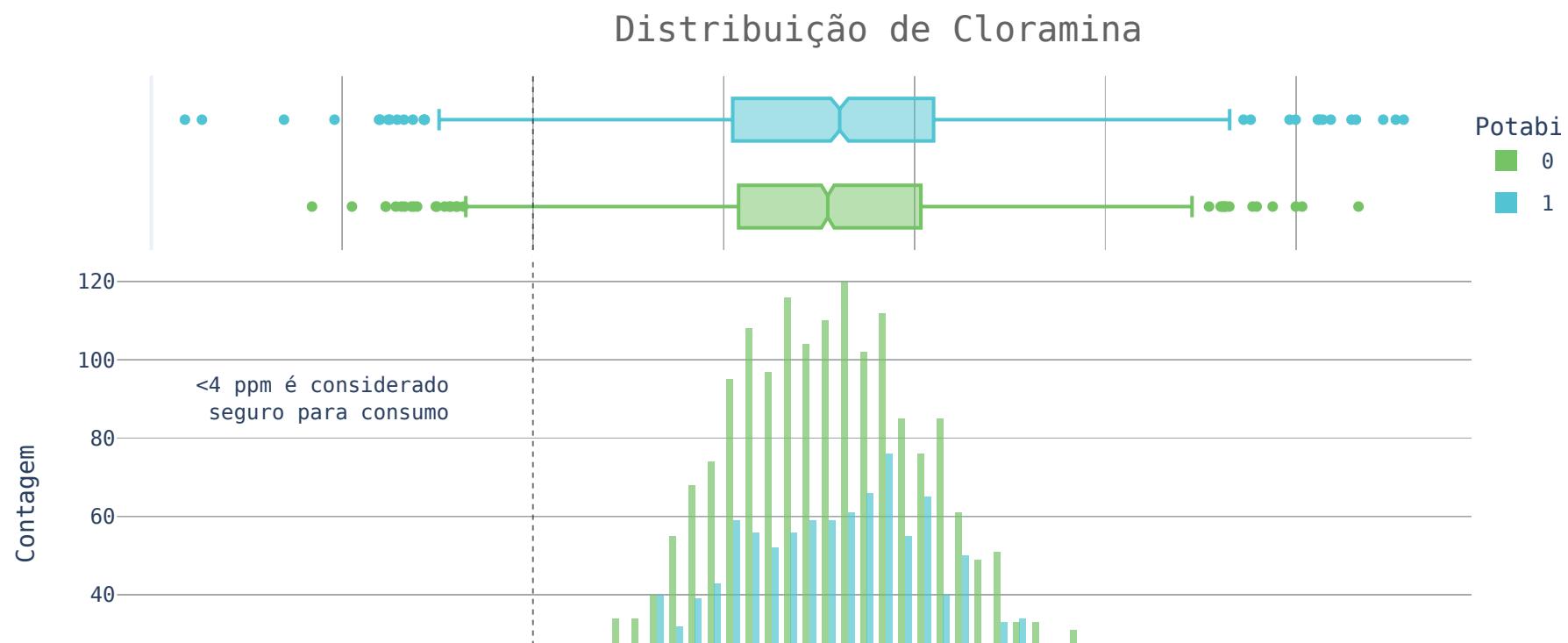
Cloraminas são uma família de subprodutos de desinfecção (DBPs), formados a partir da reação do cloro desinfetante com o átomo de nitrogênio (N) em amônia ( $\text{NH}_3$ ) ou compostos orgânicos contendo um átomo de nitrogênio reativo. Existem muitos desses produtos químicos biológicos na água potável, principalmente derivados de detritos celulares de bactérias e algas mortas. A mono-cloramina é o membro mais simples e mais comum do grupo, muitas vezes produzido intencionalmente a partir da reação de cloro puro e amônia pura.

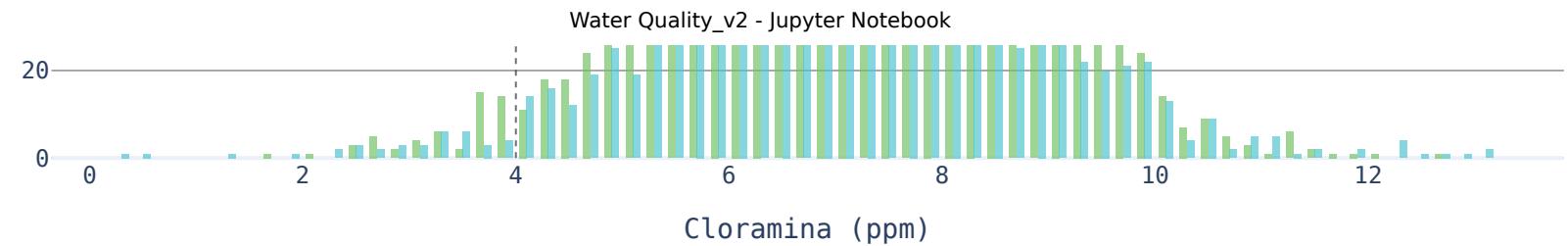
```
In [93]: fig = px.histogram(df,x='Cloraminas',y=Counter(df['Cloraminas']),color='Potabilidade',template='plotly_white'
                           marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1]],
                           barmode='group',histfunc='count')

fig.add_vline(x=4, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)

fig.add_annotation(text='<4 ppm é considerado<br> seguro para consumo',x=1.8,y=90,showarrow=False)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição de Cloramina',x=0.53,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Cloramina (ppm)',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```





## Sulfato:

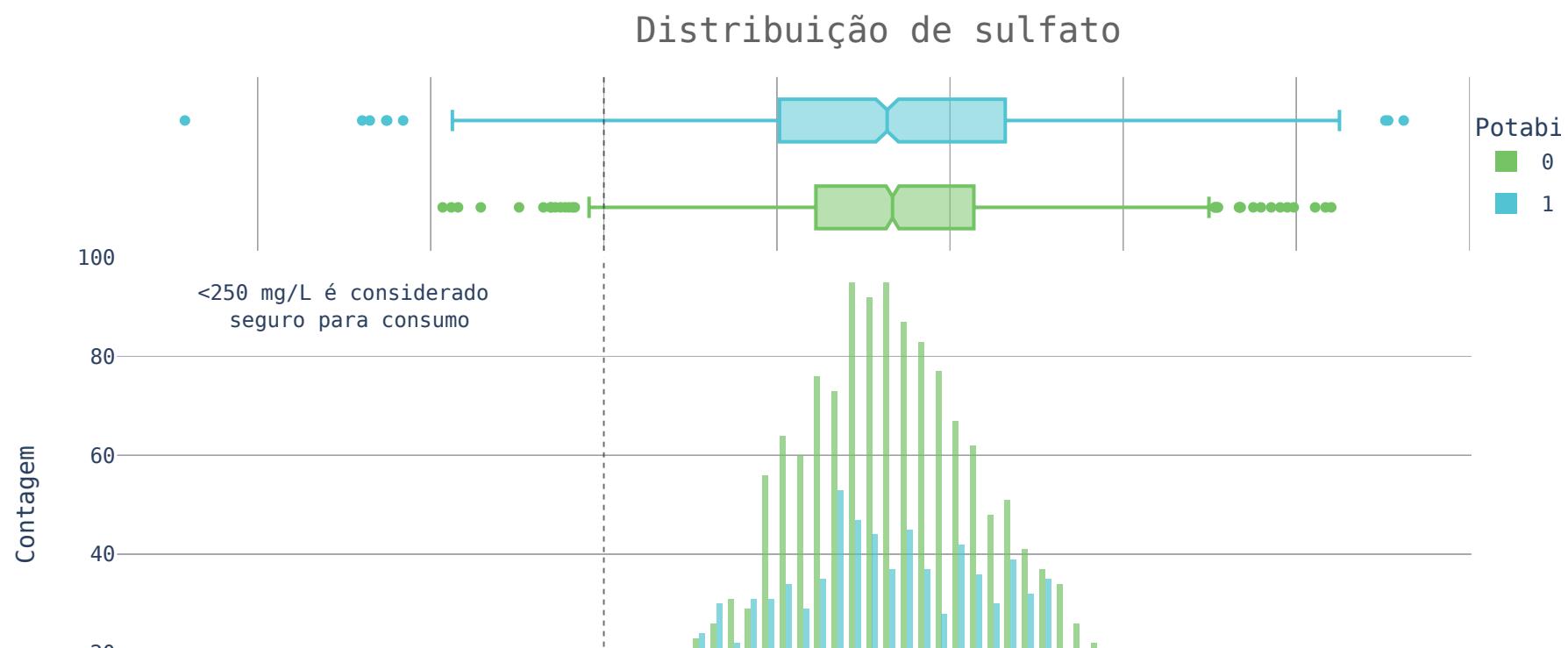
Sulfato ( $\text{SO}_4$ ) Nas águas para abastecimento público, o sulfato deve ser controlado porque provoca efeitos laxativos, sendo o padrão de potabilidade fixado em 250 mg/L pela Portaria 518 do Ministério da Saúde.

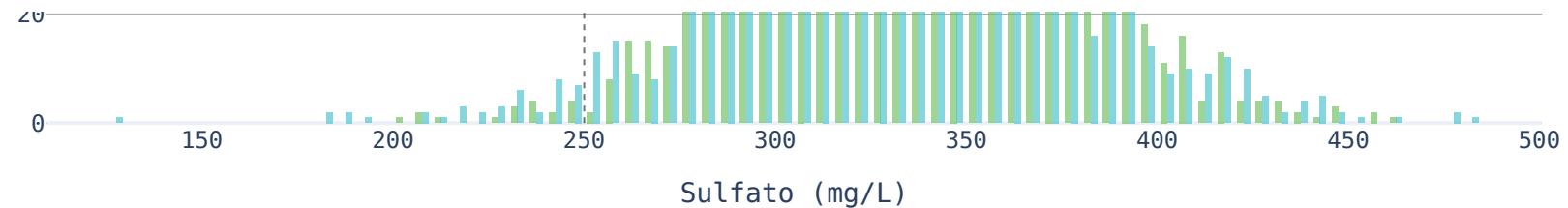
```
In [94]: fig = px.histogram(df,x='Sulfato',y=Counter(df['Sulfato']),color='Potabilidade',template='plotly_white',
                         marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1]],
                         barmode='group',histfunc='count')

fig.add_vline(x=250, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)

fig.add_annotation(text='<250 mg/L é considerado<br> seguro para consumo',x=175,y=90,showarrow=False)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição de sulfato',x=0.53,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Sulfato (mg/L)',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```





## Conductividade:

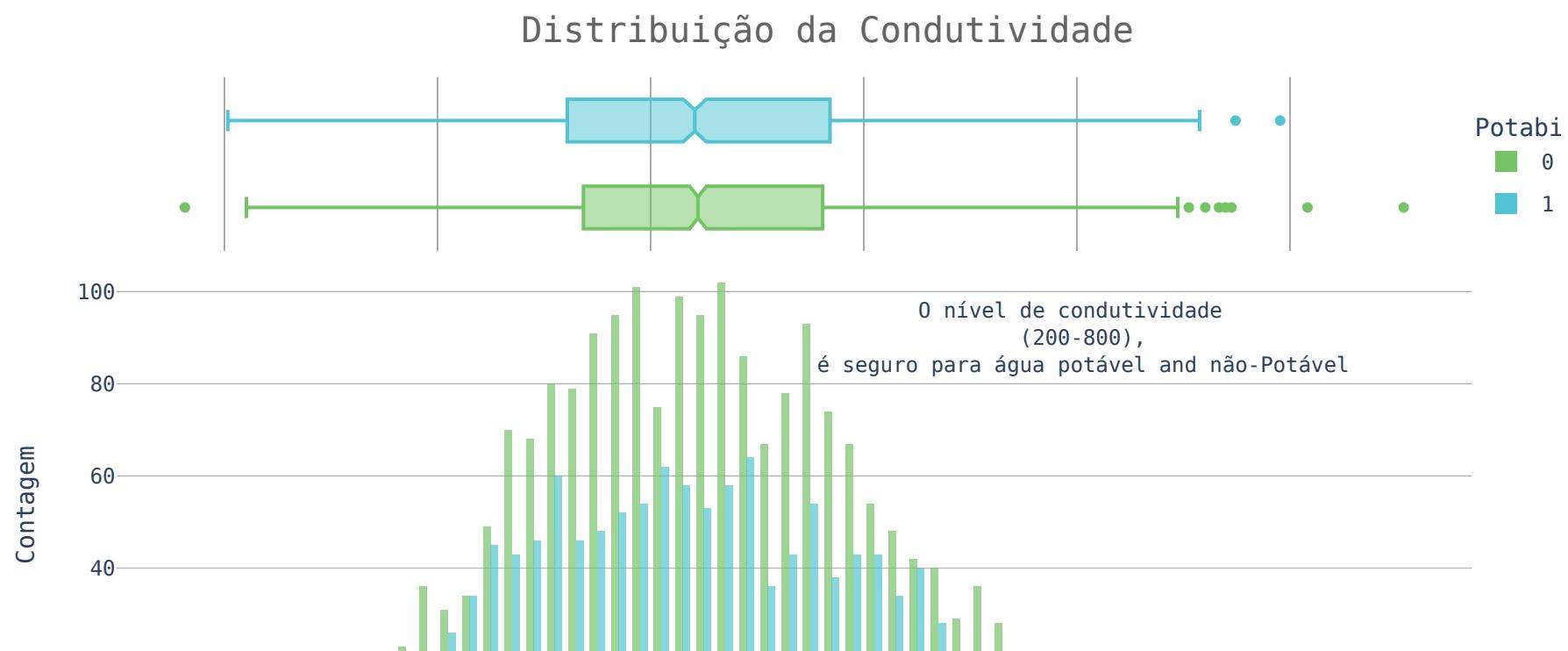
A condutividade elétrica da água é uma medida da capacidade desta em conduzir corrente elétrica, sendo proporcional à concentração de íons dissociados em um sistema aquoso. Esse parâmetro não identifica quais são os íons presentes na água, mas é um indicador importante de possíveis fontes poluidoras (ZUIN; IORIATTI; MATHEUS, 2009).

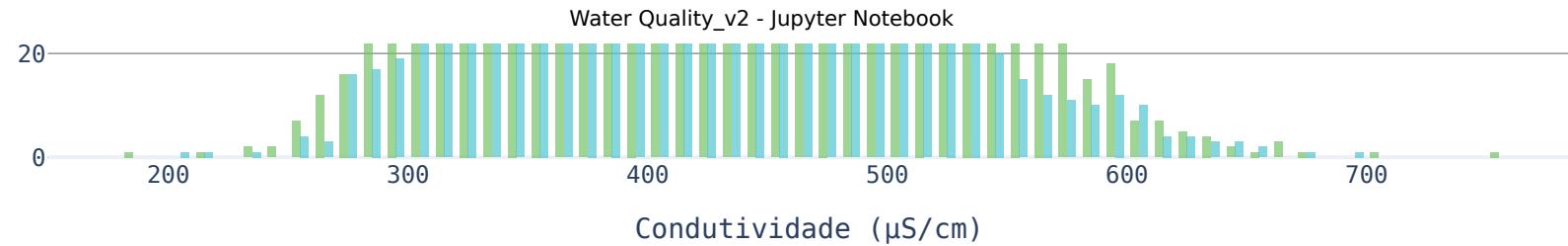
Condutividade é diretamente proporcional à temperatura da água.

```
In [95]: fig = px.histogram(df,x='Conductividade',y=Counter(df['Conductividade']),color='Potabilidade',template='plotly_mincolors',
                           marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1]],
                           barmode='group',histfunc='count')

fig.add_annotation(text='0 nível de condutividade <br> (200-800),<br> é seguro para água potável and não-Potável',
                    x=600,y=90,showarrow=False)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição da Condutividade',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Condutividade ( $\mu$ S/cm)',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```





## Carbono Orgânico:

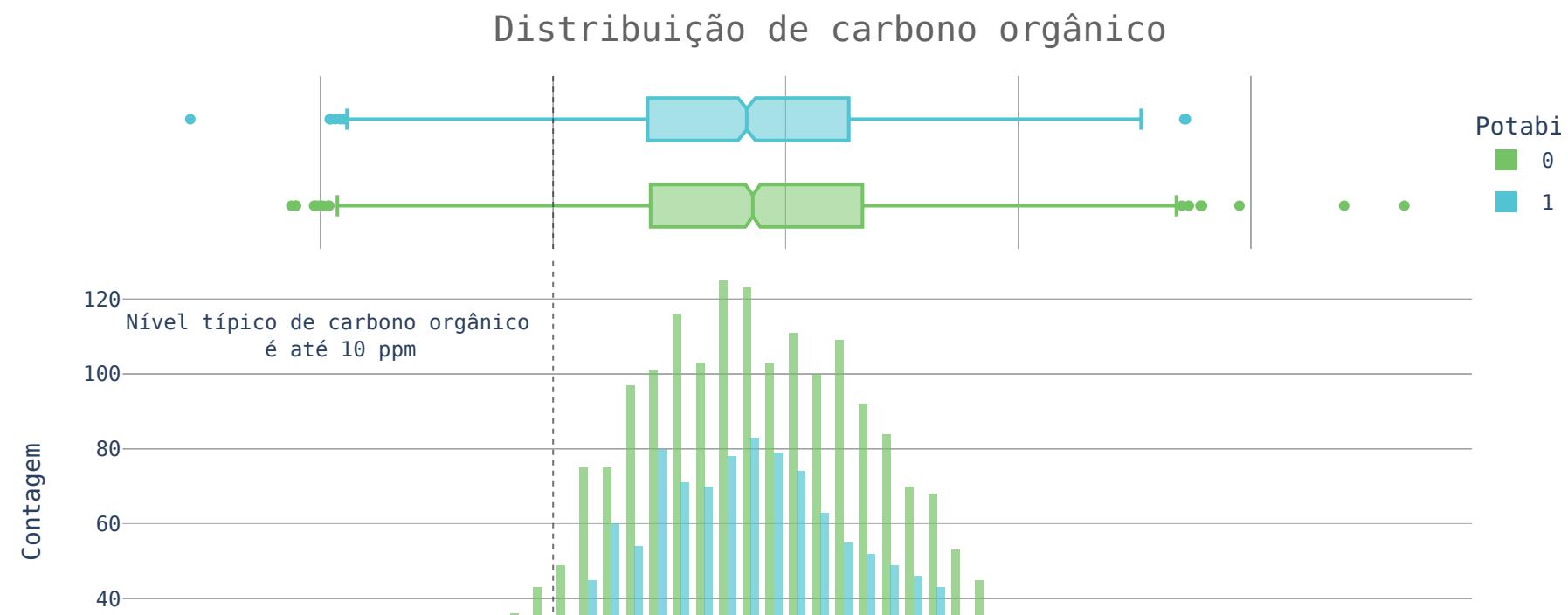
Carbono orgânico total é matéria orgânica — também conhecido como qualquer contaminante à base de carbono — na água não tratada. Pode conter milhares de partes, incluindo partículas macroscópicas, macromoléculas dissolvidas, colóides ou compostos como: substâncias orgânicas naturais, inseticidas, herbicidas, etc.

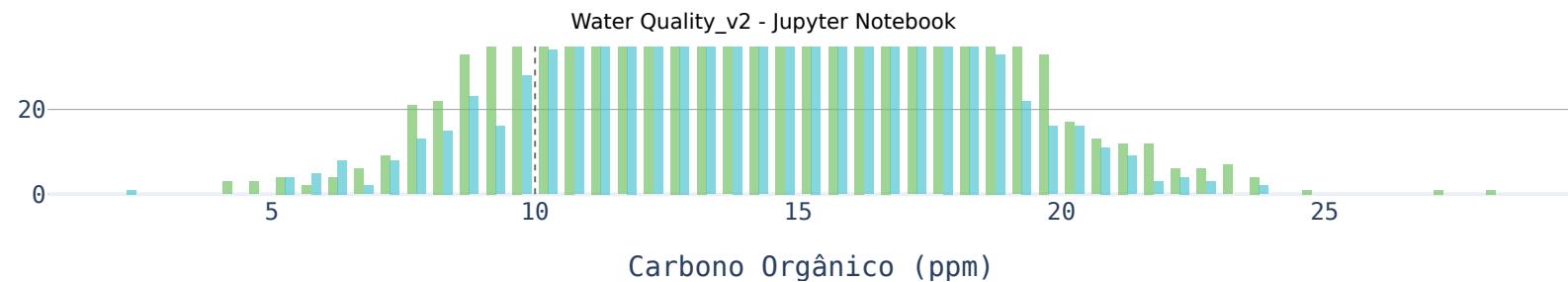
```
In [96]: fig = px.histogram(df,x='Carbono_Organico',y=Counter(df['Carbono_Organico']),color='Potabilidade',template='marginal=box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1]],barmode='group',histfunc='count')

fig.add_vline(x=10, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)

fig.add_annotation(text='Nível típico de carbono orgânico <br> é até 10 ppm',x=5.3,y=110,showarrow=False)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição de carbono orgânico',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Carbono Orgânico (ppm)',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```





## Trihalometanos:

Trihalometanos (THMs) Trihalometanos (THMs) constituem um grupo de compostos químicos e orgânicos que derivam do metano ( $\text{CH}_4$ ), em cuja molécula três de seus quatro átomos de hidrogênio foram substituídos por halogênios, isto é, átomos de cloro, bromo ou iodo. Eles se formam durante o processo de desinfecção das águas destinadas ao consumo humano e podem causar diversos danos à saúde, como problemas no sistema reprodutivo, abortos espontâneos e maior propensão ao câncer. A formação de trihalometanos em função da cloração da água é conhecida desde 1974 e por isso controlada. A Portaria n.º 36 de 19.01.90, do Ministério da Saúde, que regulamenta a qualidade de água destinada ao consumo humano, limita o teor de trihalometanos em 100 microgramas/litro.

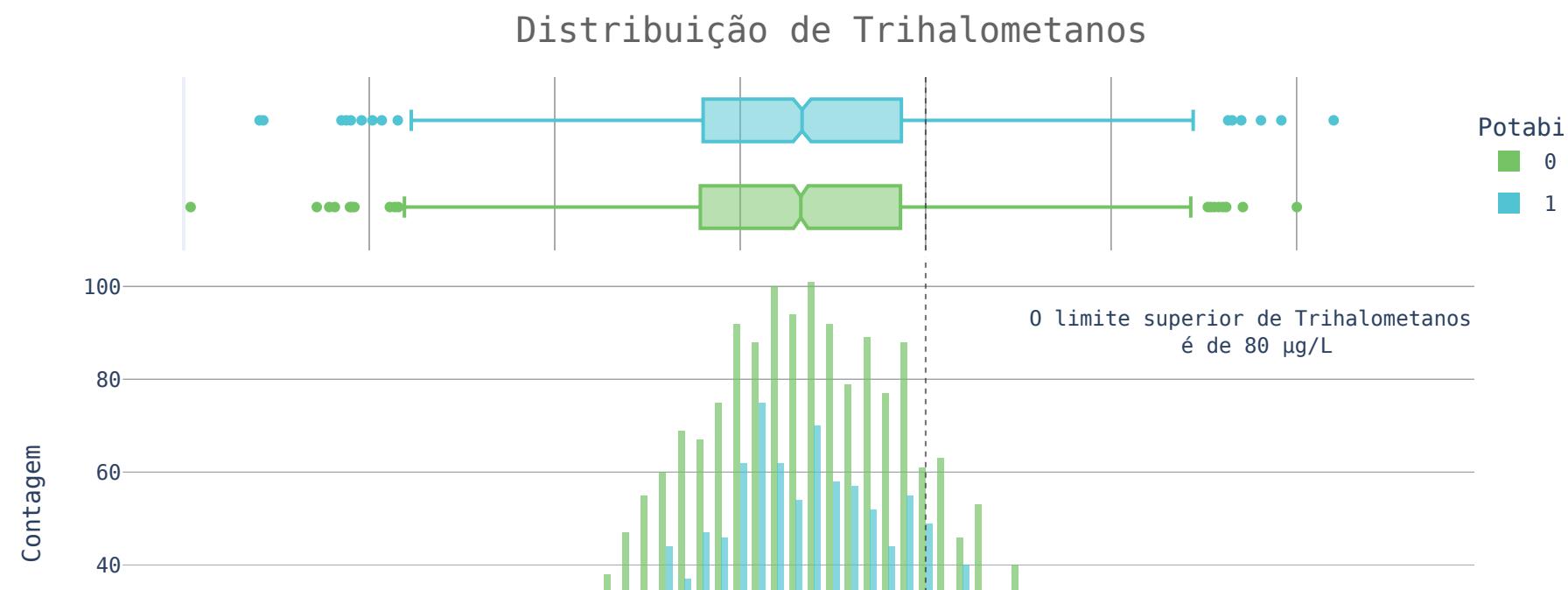
In [97]:

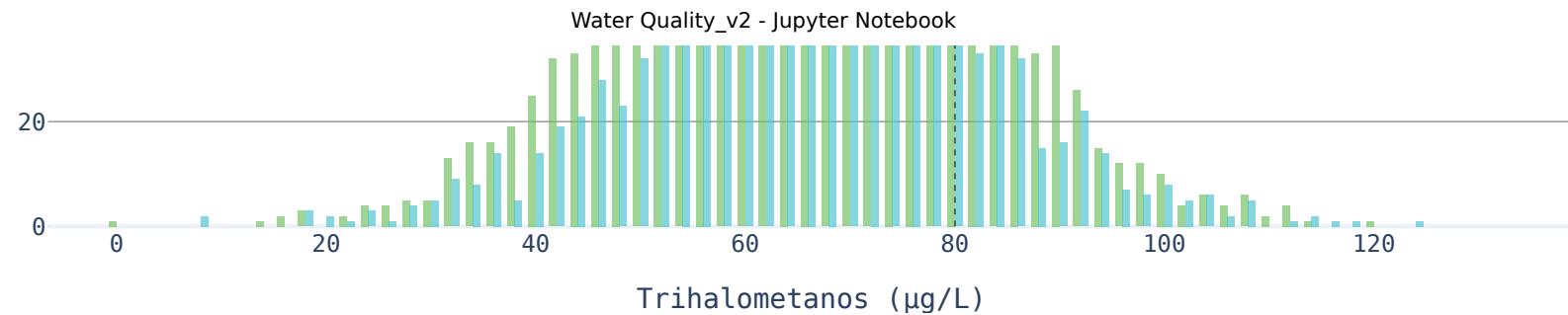
```
fig = px.histogram(df,x='Trihalometanos',y=Counter(df['Trihalometanos']),color='Potabilidade',template='plotly_mincolors',
                    marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1]],
                    barmode='group',histfunc='count')

fig.add_vline(x=80, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)

fig.add_annotation(text='O limite superior de Trihalometanos<br>é de 80 µg/L',x=115,y=90,showarrow=False)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição de Trihalometanos',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Trihalometanos (µg/L)',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```





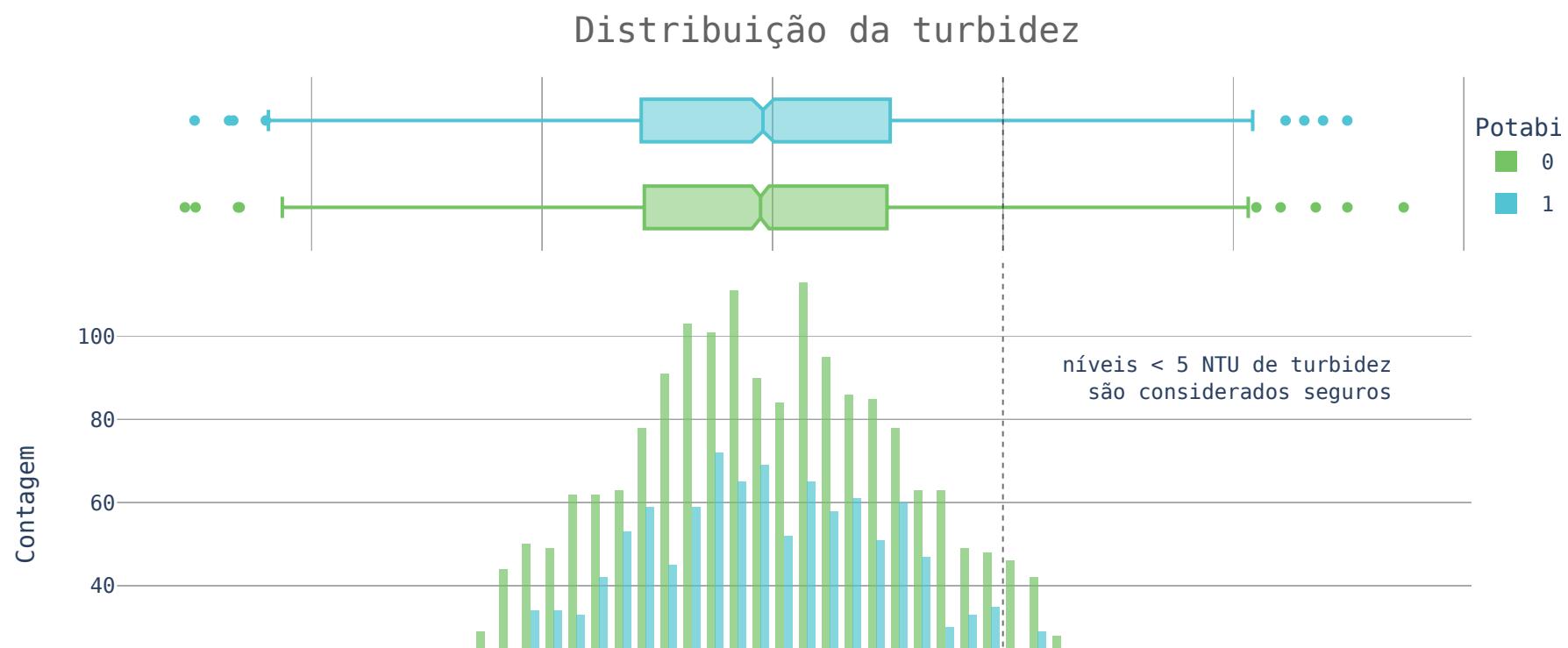
Turbidez: É a medição da resistência da água à passagem de luz. É provocada pela presença de partículas flutuando na água. A turbidez é um parâmetro de aspecto estético de aceitação ou rejeição do produto, e o valor máximo permitido de turbidez na água distribuída é de 5,0 NTU.

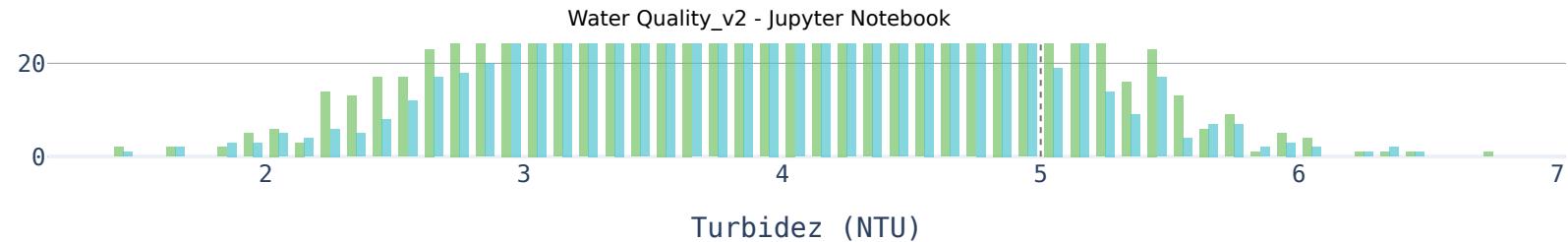
```
In [98]: fig = px.histogram(df,x='Turbidez',y=Counter(df['Turbidez']),color='Potabilidade',template='plotly_white',
                         marginal='box',opacity=0.7,nbins=100,color_discrete_sequence=[colors_green[3],colors_blue[1]],
                         barmode='group',histfunc='count')

fig.add_vline(x=5, line_width=1, line_color=colors_dark[1],line_dash='dot',opacity=0.7)

fig.add_annotation(text='níveis < 5 NTU de turbidez <br> são considerados seguros',x=6,y=90,showarrow=False)

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribuição da turbidez',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Turbidez (NTU)',
    yaxis_title_text='Contagem',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```

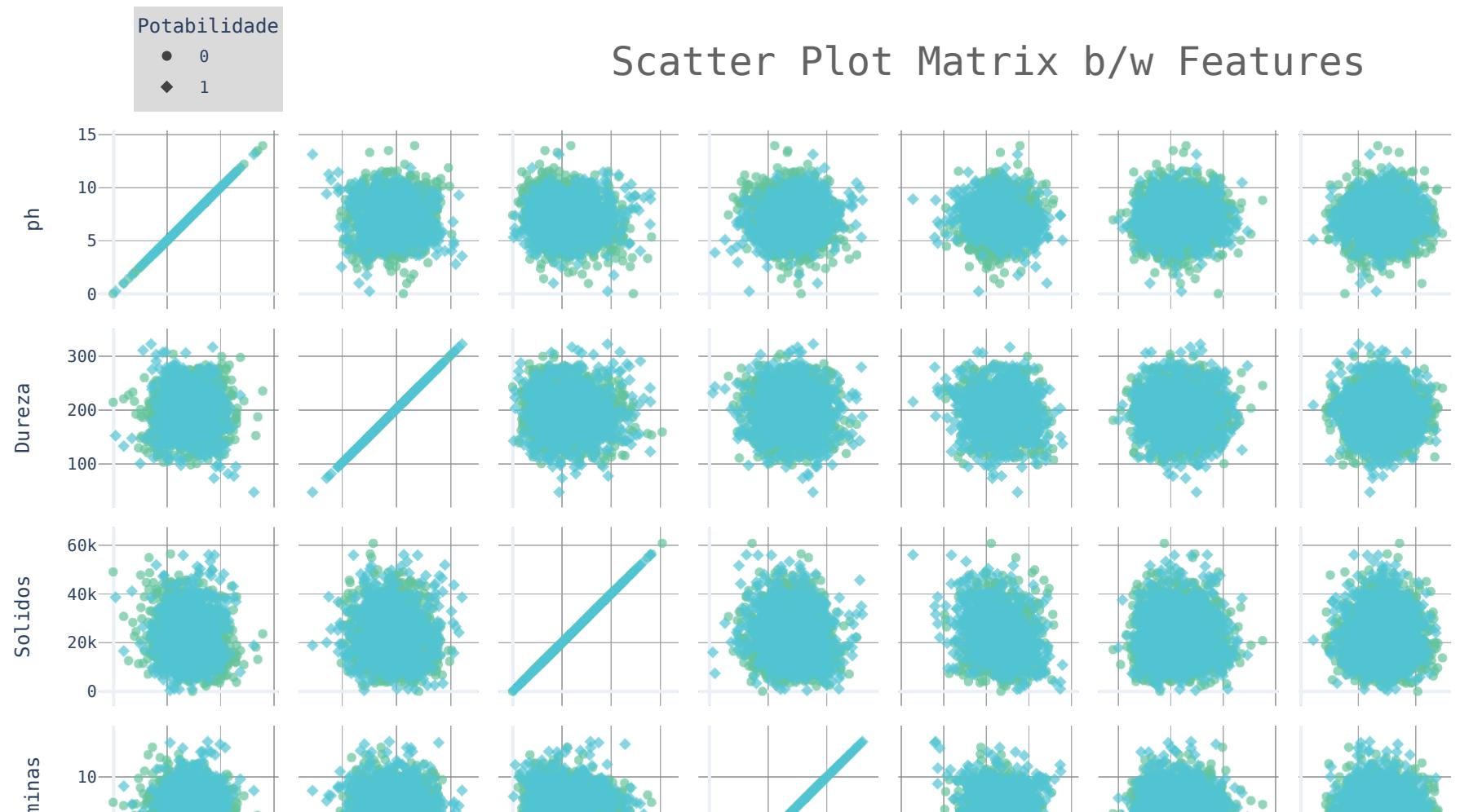


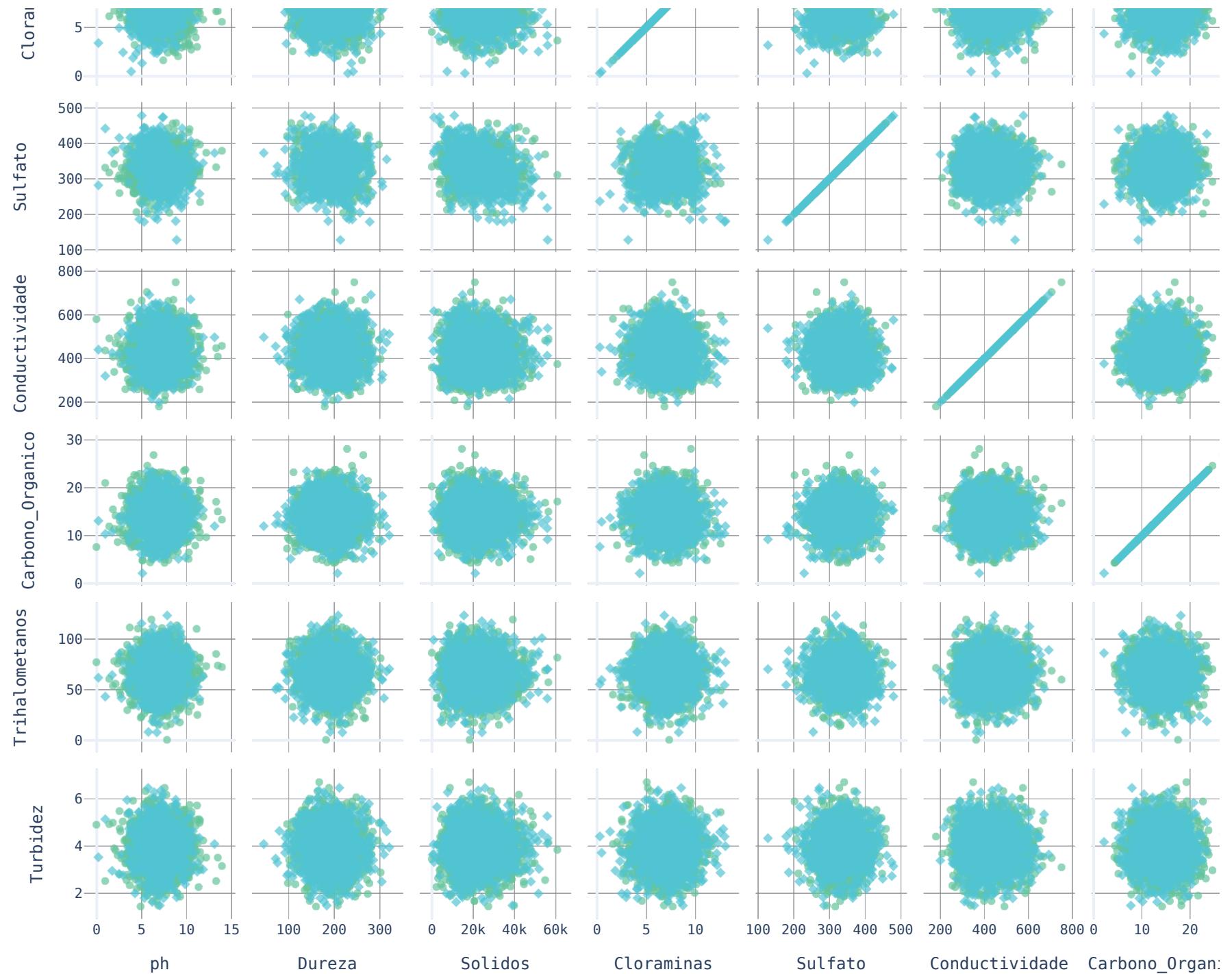


## Scatter Plot Matrix ajuda-nos a encontrar a correlação entre as todas as features

Na matriz abaixo podemos visualizar dados bivariados que são o resultado da observação de duas variáveis sobre o mesmo indivíduo da amostra. Como as variáveis podem depender ou não uma da outra, o que chamamos de correlação. No gráfico abaixo, podemos observar que não há uma dispersão muito forte entre todas as variáveis verificadas duas a duas para inferir a contribuição para agua ser potável ou não. Em outras palavras, pouca relação entre as features.

```
In [99]: fig = px.scatter_matrix(df, df.drop('Potabilidade', axis=1), height=1250, width=1250, template='plotly_white', opacity=0.8, color_discrete_sequence=[colors_blue[3], colors_green[3]], color='Potabilidade', symbol='Potabilidade', color_continuous_scale=[colors_green[3], colors_blue[3]])  
fig.update_layout(font_family='monospace', font_size=10,  
                  coloraxis_showscale=False,  
                  legend=dict(x=0.02, y=1.07, bgcolor=colors_dark[4]),  
                  title=dict(text='Scatter Plot Matrix b/w Features', x=0.5, y=0.97,  
                             font=dict(color=colors_dark[2], size=24)))  
fig.show()
```





```
In [100]: cor=df.drop('Potabilidade',axis=1).corr()  
cor
```

Out[100]:

	ph	Dureza	Solidos	Cloraminas	Sulfato	Conductividade	Carbono_Organico	Trihalometanos	Turbidez
ph	1.000000	0.082096	-0.089288	-0.034350	0.018203	0.018614	0.043503	0.003354	-0.039057
Dureza	0.082096	1.000000	-0.046899	-0.030054	-0.106923	-0.023915	0.003610	-0.013013	-0.014449
Solidos	-0.089288	-0.046899	1.000000	-0.070148	-0.171804	0.013831	0.010242	-0.009143	0.019546
Cloraminas	-0.034350	-0.030054	-0.070148	1.000000	0.027244	-0.020486	-0.012653	0.017084	0.002363
Sulfato	0.018203	-0.106923	-0.171804	0.027244	1.000000	-0.016121	0.030831	-0.030274	-0.011187
Conductividade	0.018614	-0.023915	0.013831	-0.020486	-0.016121	1.000000	0.020966	0.001285	0.005798
Carbono_Organico	0.043503	0.003610	0.010242	-0.012653	0.030831	0.020966	1.000000	-0.013274	-0.027308
Trihalometanos	0.003354	-0.013013	-0.009143	0.017084	-0.030274	0.001285	-0.013274	1.000000	-0.022145
Turbidez	-0.039057	-0.014449	0.019546	0.002363	-0.011187	0.005798	-0.027308	-0.022145	1.000000

## Heatmap

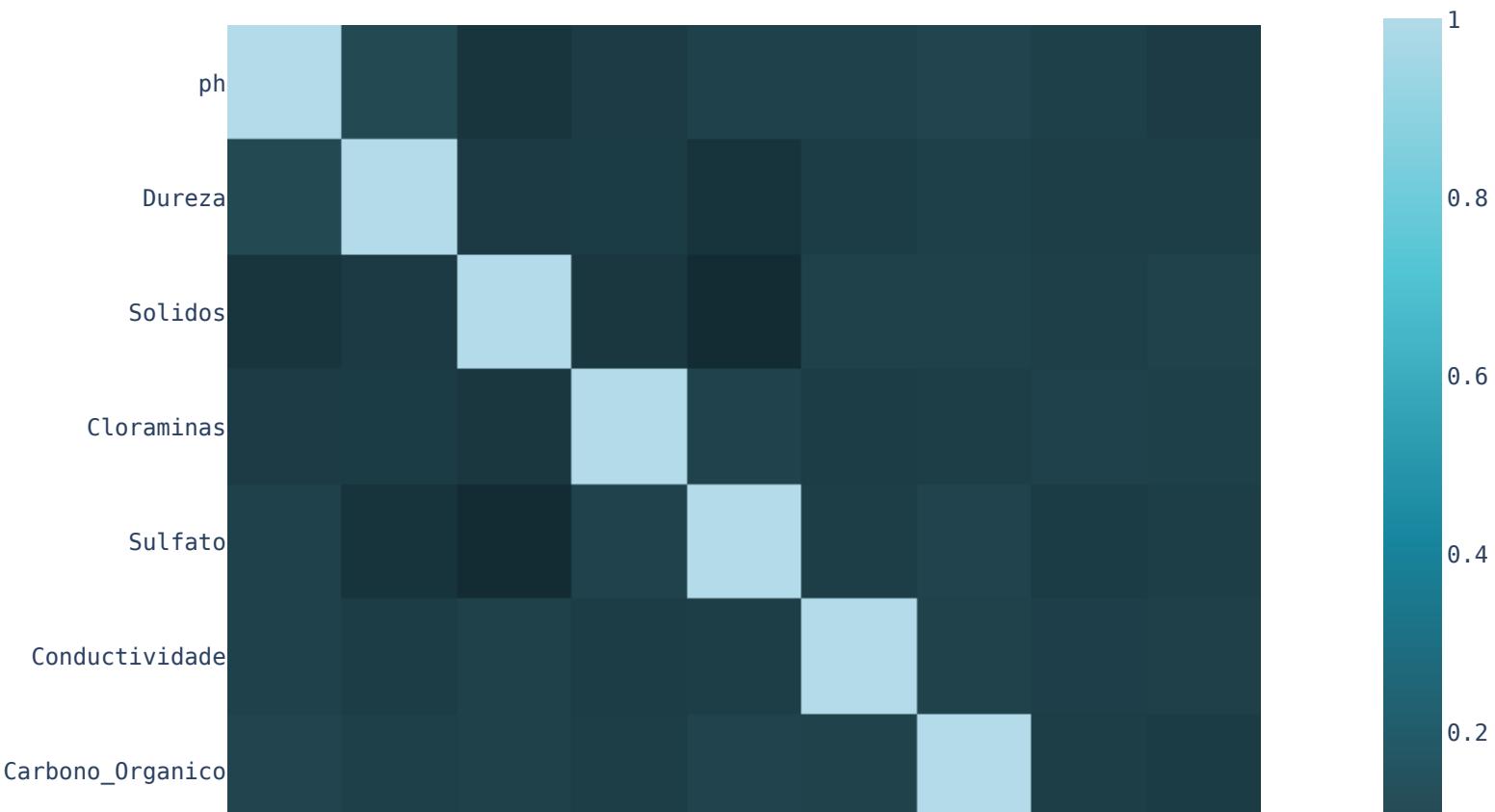
Vamos visualizar o a correlação entre as variáveis através do Heatmap. Uma correlação entre duas variáveis (linha, coluna) pode ir de -1 a 1. Valores perto de zero indicam que não há tendência linear entre as duas variáveis no ente o valores próximo a 1 mostram que a correlação é positiva, ou seja mais correlacionadas estão no sentido de que se uma aumenta em quantidade a outra também seguirá essa tendência. No lado oposto, próximo de -1 nos diz que estão fortemente relacionadas, porém, quando uma variável aumenta seu valor, a outra tende a ter seu valor diminuído.

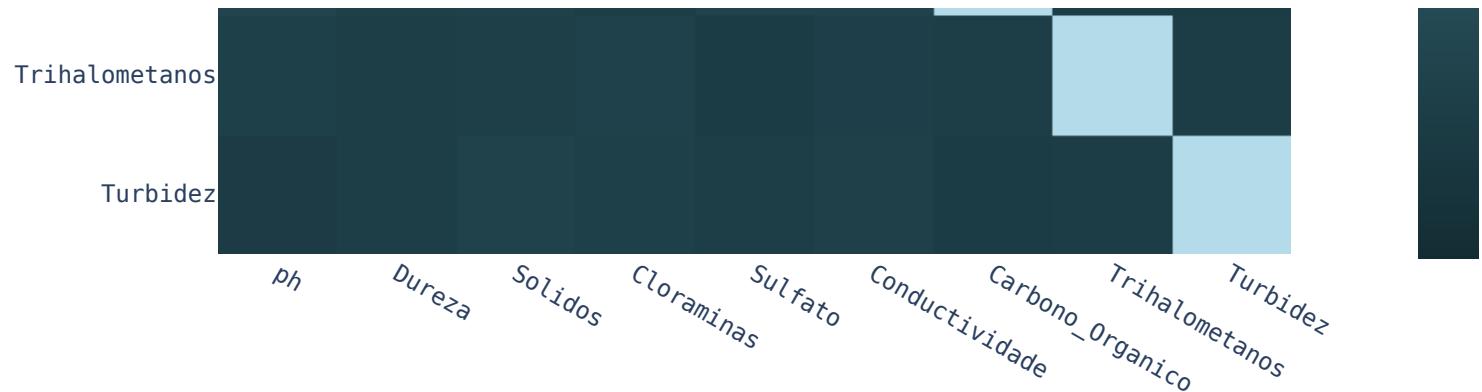
```
In [101]: fig = px.imshow(cor,height=800,width=800,color_continuous_scale=colors_blue,template='plotly_white')

fig.update_layout(font_family='monospace',
                  title=dict(text='Correlation Heatmap',x=0.5,y=0.93,
                             font=dict(color=colors_dark[2],size=24)),
                  coloraxis_colorbar=dict(len=0.85,x=1.1)
                 )

fig.show()
```

## Correlation Heatmap



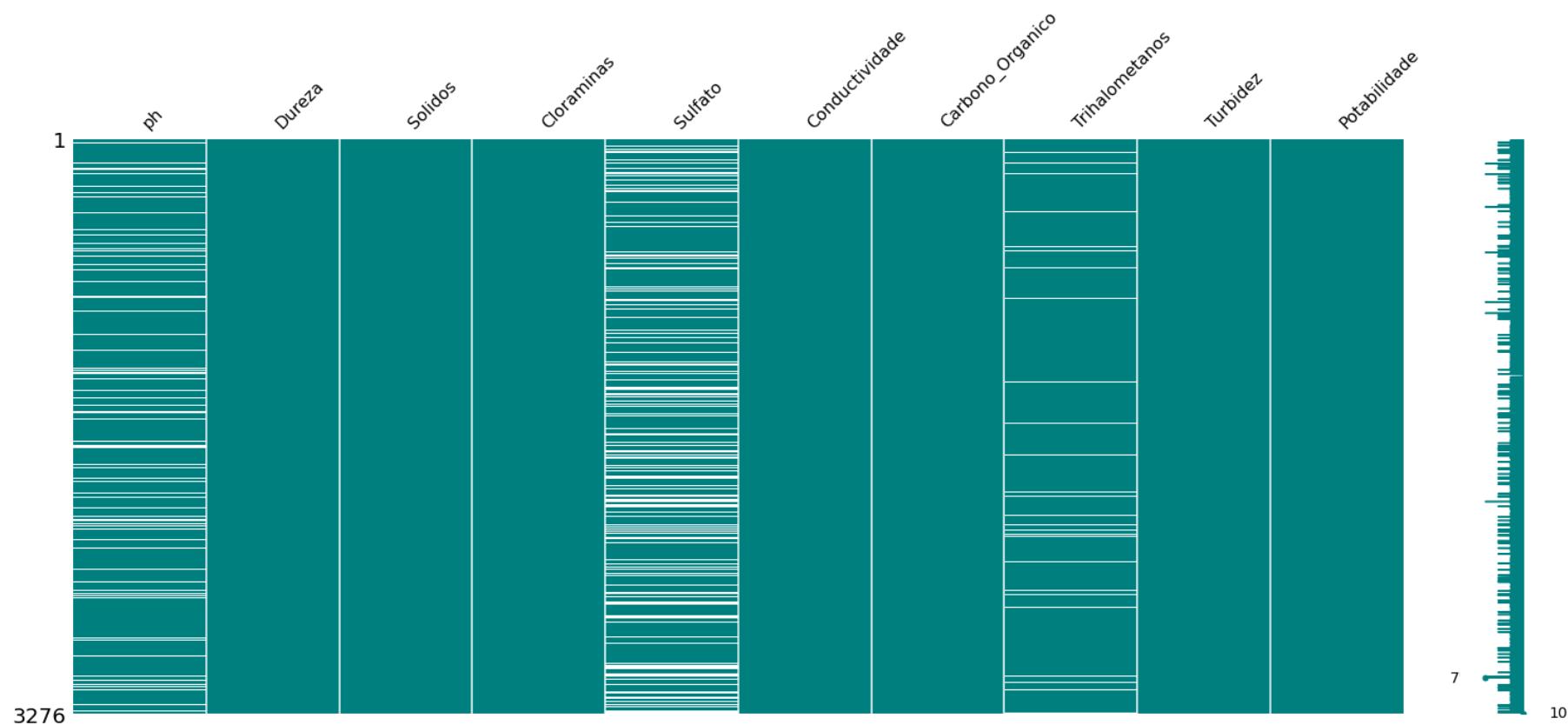


## Visualizando dados faltantes

É bem comum que alguns valores em datasets estejam faltando. E essa falta é representada por um valor convencionando de NaN (Not a Number). Para se ter um entendimento de como está nossos dados podemos utilizar a biblioteca Missingno Library.

Na matriz abaixo podemos ter uma ideia da quantidade de dados faltantes em cada coluna antes de fazer um tratamento de dados, de fato. Por exemplo, a coluna -Sulfato é a que possui o maior número de dados faltantes.

```
In [102]: fig = msno.matrix(df,color=(0,0.5,0.5))
```



We can see that there are many missing values in the dataset, so we'll try to deal with it

## Tratamento dos dados

Como podemos ver abaixo existem vários valores `NaN`, *not a number*, na base. Portanto precisaremos tratar os dados, e removendo os valores nulos. Depois realizamos uma técnica conhecida como *undersampling*, na qual igualamos o número de instâncias das duas classes removendo as instâncias da classe com mais valores.

```
In [103]: df.isnull().sum()
```

```
Out[103]: ph           491
Dureza          0
Solidos          0
Cloraminas       0
Sulfato          781
Conductividade   0
Carbono_Organico 0
Trihalometanos    162
Turbidez          0
Potabilidade      0
dtype: int64
```

In [104]: df[df['Potabilidade']==0].describe()

Out[104]:

	ph	Dureza	Solidos	Cloraminas	Sulfato	Conductividade	Carbono_Organico	Trihalometanos	Turbidez	Potal
count	1684.000000	1998.000000	1998.000000	1998.000000	1510.000000	1998.000000	1998.000000	1891.000000	1998.000000	
mean	7.085378	196.733292	21777.490788	7.092175	334.564290	426.730454	14.364335	66.303555	3.965800	
std	1.683499	31.057540	8543.068788	1.501045	36.745549	80.047317	3.334554	16.079320	0.780282	
min	0.000000	98.452931	320.942611	1.683993	203.444521	181.483754	4.371899	0.738000	1.450000	
25%	6.037723	177.823265	15663.057382	6.155640	311.264006	368.498530	12.101057	55.706530	3.444062	
50%	7.035456	197.123423	20809.618280	7.090334	333.389426	422.229331	14.293508	66.542198	3.948076	
75%	8.155510	216.120687	27006.249009	8.066462	356.853897	480.677198	16.649485	77.277704	4.496106	
max	14.000000	304.235912	61227.196008	12.653362	460.107069	753.342620	28.300000	120.030077	6.739000	

In [105]: df[df['Potabilidade']==1].describe()

Out[105]:

	ph	Dureza	Solidos	Cloraminas	Sulfato	Conductividade	Carbono_Organico	Trihalometanos	Turbidez	Potabi
count	1101.000000	1278.000000	1278.000000	1278.000000	985.000000	1278.000000	1278.000000	1223.000000	1278.000000	
mean	7.073783	195.800744	22383.991018	7.169338	332.566990	425.383800	14.160893	66.539684	3.968328	
std	1.448048	35.547041	9101.010208	1.702988	47.692818	82.048446	3.263907	16.327419	0.780842	
min	0.227499	47.432000	728.750830	0.352000	129.000000	201.619737	2.200000	8.175876	1.492207	
25%	6.179312	174.330531	15668.985035	6.094134	300.763772	360.939023	12.033897	56.014249	3.430909	
50%	7.036752	196.632907	21199.386614	7.215163	331.838167	420.712729	14.162809	66.678214	3.958576	
75%	7.933068	218.003420	27973.236446	8.199261	365.941346	484.155911	16.356245	77.380975	4.509569	
max	13.175402	323.124000	56488.672413	13.127000	481.030642	695.369528	23.604298	124.000000	6.494249	

```
In [106]: df[df['Potabilidade']==0][['ph','Sulfato','Trihalometanos']].median()
```

```
Out[106]: ph           7.035456  
Sulfato        333.389426  
Trihalometanos 66.542198  
dtype: float64
```

Podemos notar que a diferença entre a média e a mediana de valores para água potável e não potável é mínima. Então nós usamos a mediana geral, ou seja, o valor central da feature como a entrada.

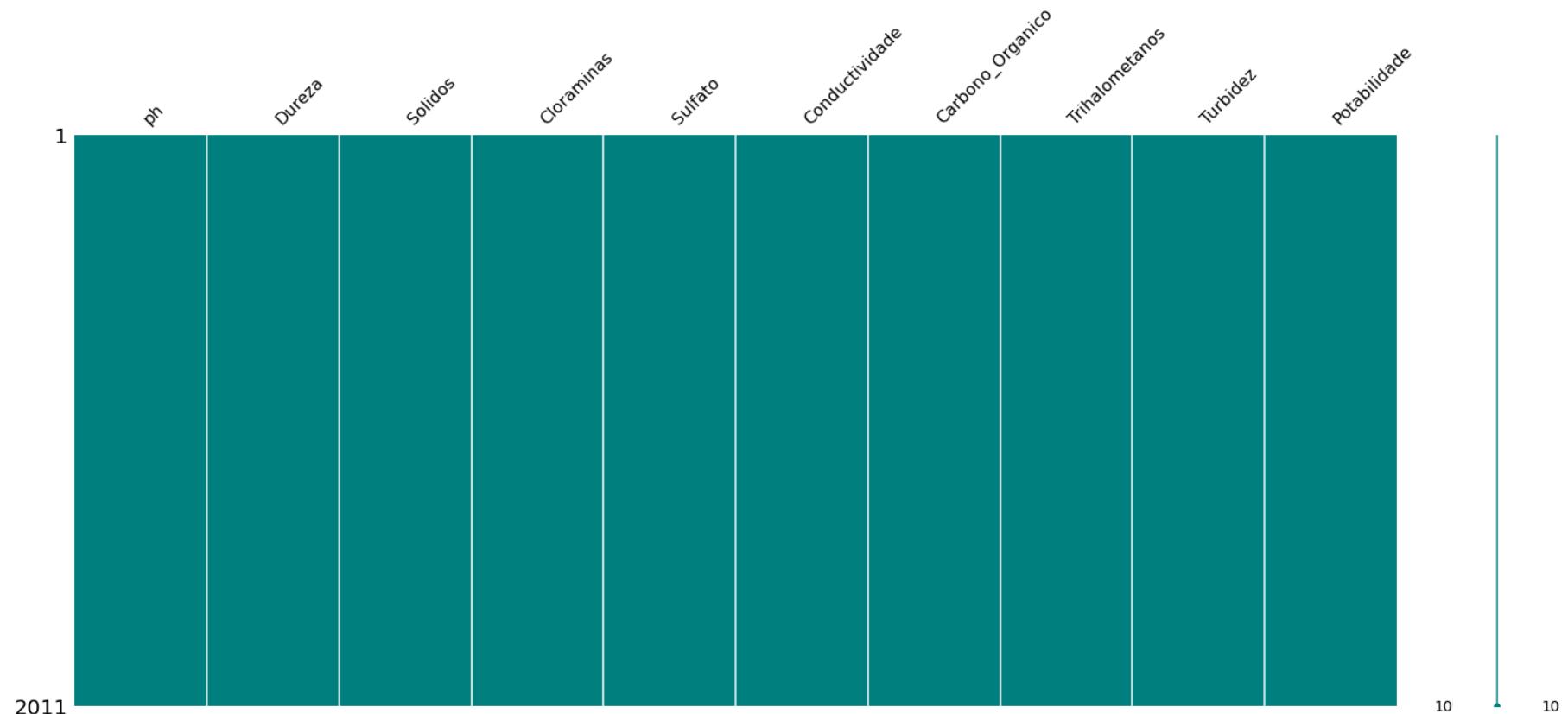
```
In [107]: df = df.dropna()  
df.isnull().sum()
```

```
Out[107]: ph          0  
Dureza        0  
Solidos        0  
Cloraminas     0  
Sulfato         0  
Conductividade 0  
Carbono_Organico 0  
Trihalometanos 0  
Turbidez        0  
Potabilidade    0  
dtype: int64
```

## Visualizando dados faltantes

Após o tratamento não há mais NaN entre os valores das amostras.

```
In [108]: fig = msno.matrix(df,color=(0,0.5,0.5))
```



Os dados foram tratados para terem 50% de dados de água potável e 50% de dados de água não potável. Como veremos a seguir, pelos resultados.

```
In [109]: import numpy as np

not_potable_indices = df[df.Potabilidade == 0].index
number_of_potables = df[df.Potabilidade == 1].shape[0]
random_indices = np.random.choice(not_potable_indices, number_of_potables, replace=False)
df_not_potables = df.loc[random_indices]
df_potables = df[df.Potabilidade == 1]
df = pd.concat([df_potables, df_not_potables])

df.Potabilidade.value_counts()
```

```
Out[109]: 0    811
           1    811
Name: Potabilidade, dtype: int64
```

## Dividindo os dados em treino e teste

Dividimos os dados em  $X$ , sendo as instâncias e  $y$  os valores da classe "Potabilidade".

```
In [110]: from sklearn.model_selection import train_test_split

# X são as instâncias da (uma linha) base de dados, sem as classes (Potabilidade)
X = df.drop('Potabilidade', axis=1).values

# Y são as classes de cada instâncias
y = df['Potabilidade'].values

# Divisão entre X de treino, y de treino, X de teste e y de teste.
# 30% dos dados são teste, 70% são treino
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0xB33F)
```

## Treinos e testes

### Bernoulli Naive Bayes

```
In [111]: from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import classification_report, accuracy_score

# Cria o modelo
BNB = BernoulliNB()

# Faz o treino
BNB.fit(X_train, y_train)

# Faz o teste
y_predicted = BNB.predict(X_test)

# Plota métricas
target_names = ['Não Potável', 'Potável']
print(f"Accuracy: {accuracy_score(y_test, y_predicted)}")
print(classification_report(y_test, y_predicted, target_names=target_names))
```

```
Accuracy: 0.4804928131416838
      precision    recall  f1-score   support
  Não Potável       0.48     1.00     0.65      234
  Potável          0.00     0.00     0.00      253

  accuracy         0.48      --      487
  macro avg        0.24     0.50     0.32      487
  weighted avg     0.23     0.48     0.31      487
```

```
/home/aritana/my_jupyter_notebook/my_jupyter_notebook/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1248: UndefinedMetricWarning:
```

```
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

```
/home/aritana/my_jupyter_notebook/my_jupyter_notebook/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1248: UndefinedMetricWarning:
```

```
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

```
/home/aritana/my_jupyter_notebook/my_jupyter_notebook/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1248: UndefinedMetricWarning:  
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

## Matriz confusao

		Predicted class	
		<i>P</i>	<i>N</i>
Actual Class	<i>P</i>	True Positives (TP)	False Negatives (FN)
	<i>N</i>	False Positives (FP)	True Negatives (TN)

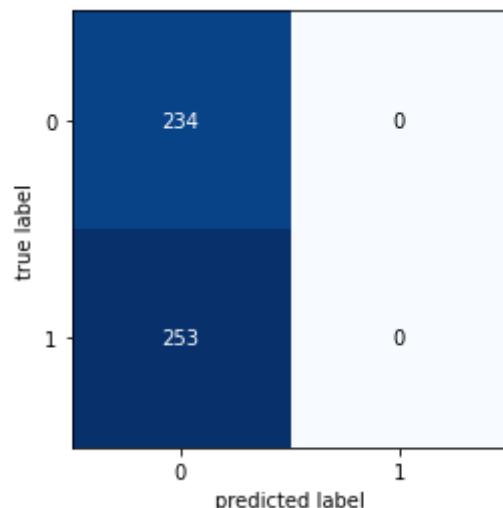
```
In [112]: from sklearn.metrics import confusion_matrix  
print (confusion_matrix(y_test, y_predicted))
```

```
[[234  0]  
 [253  0]]
```

```
In [113]: from mlxtend.plotting import plot_confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

binary1 = confusion_matrix(y_test, y_predicted)

fig, ax = plot_confusion_matrix(conf_mat=binary1)
plt.show()
```



**Queremos prever a classe água potável e da tabela acima podemos inferir:**

- TP - 234. Número de vezes que a classe que estamos buscando foi prevista corretamente.
- FN - 0 - Número de vezes que a classe que não estamos buscando foi prevista corretamente.
- FP - 253 - Número de vezes que a classe que estamos buscando foi prevista incorretamente.
- TN - 0 - Número de vezes que a classe que não estamos buscando foi prevista incorretamente.

<https://medium.com/data-hackers/entendendo-o-que-%C3%A9-matriz-de-confus%C3%A3o-com-python-114e683ec509>  
[\(https://medium.com/data-hackers/entendendo-o-que-%C3%A9-matriz-de-confus%C3%A3o-com-python-114e683ec509\)](https://medium.com/data-hackers/entendendo-o-que-%C3%A9-matriz-de-confus%C3%A3o-com-python-114e683ec509)

## Análise

É interessante notar que o Bernoulli Naive Bayes não conseguiu generalizar bem, classificando todos as instâncias como 0, não potáveis. Aqui nós ainda temos que descobrir o porquê (ou não) do que isso aconteceu pra colocar no artigo. Como não existem praticamente nenhum parâmetro

## Random Forest - Conjunto de parâmetros 1

```
In [114]: from sklearn.ensemble import RandomForestClassifier

# Cria o modelo
RF = RandomForestClassifier(n_estimators=100, min_samples_leaf=2, random_state=0xB33F)

# Faz o treino
RF.fit(X_train, y_train)

# Faz o teste
y_predicted = RF.predict(X_test)

# Plota métricas
target_names = ['Not Potable', 'Potable']
print(f"Accuracy: {accuracy_score(y_test, y_predicted)}")
print(classification_report(y_test, y_predicted, target_names=target_names))
```

Accuracy: 0.5975359342915811  

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

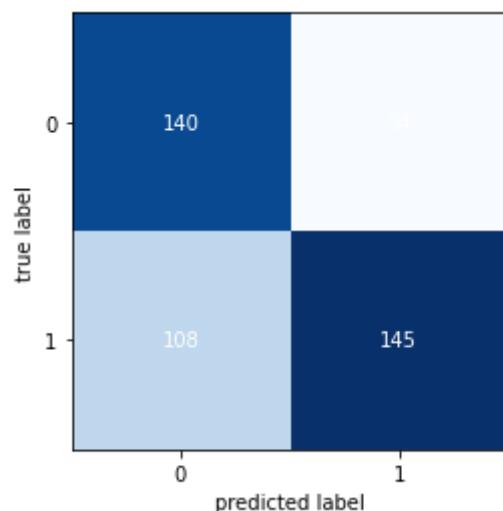
Not Potable	0.58	0.59	0.59	234
Potable	0.62	0.60	0.61	253
accuracy			0.60	487
macro avg	0.60	0.60	0.60	487
weighted avg	0.60	0.60	0.60	487

## Matriz Confusao

```
In [123]: from sklearn.metrics import confusion_matrix  
print (confusion_matrix(y_test, y_predicted))
```

```
[[140  94]  
 [108 145]]
```

```
In [125]: from mlxtend.plotting import plot_confusion_matrix  
import matplotlib.pyplot as plt  
import numpy as np  
  
binary2= confusion_matrix(y_test, y_predicted)  
  
fig, ax = plot_confusion_matrix(conf_mat=binary2)  
plt.show()
```



Queremos prever a classe água potável e da tabela acima podemos inferir:

- TP - 140. Número de vezes que a classe que estamos buscando foi prevista corretamente.
- FN - 0- Número de vezes que a classe que não estamos buscando foi prevista corretamente.
- FP - 108 - Número de vezes que a classe que estamos buscando foi prevista incorretamente.

- TN - 145 - Número de vezes que a classe que não estamos buscando foi prevista incorretamente.

## Análise

Ao contrário do Bernoulli Naive Bayes, o Random Forest Classifier conseguiu generalizar para as duas classes

## Random Forest - Conjunto de parâmetros 2

```
In [117]: RF = RandomForestClassifier(n_estimators=1000, min_samples_leaf=2, random_state=0xB33F)
RF.fit(X_train, y_train)

y_predicted = RF.predict(X_test)

target_names = ['Not Potable', 'Potable']
print(f"Accuracy: {accuracy_score(y_test, y_predicted)}")
print(classification_report(y_test, y_predicted, target_names=target_names))
```

```
Accuracy: 0.5934291581108829
      precision    recall  f1-score   support
Not Potable       0.57      0.62      0.59      234
  Potable        0.62      0.57      0.59      253

           accuracy                           0.59      487
      macro avg       0.59      0.59      0.59      487
weighted avg       0.60      0.59      0.59      487
```

## Análise

Como podemos ver, o número de árvores da flores não influencia tanto na melhoria do modelo. Com 10 vezes o número maior de árvores, a melhor no *f1-score* é de apenas 0.01.

## Random Forest - Conjunto de parâmetros 3

```
In [118]: RF = RandomForestClassifier(n_estimators=100, min_samples_leaf=1, random_state=0xB33F)
RF.fit(X_train, y_train)

y_predicted = RF.predict(X_test)

target_names = ['Not Potable', 'Potable']
print(f"Accuracy: {accuracy_score(y_test, y_predicted)}")
print(classification_report(y_test, y_predicted, target_names=target_names))
```

Accuracy: 0.5852156057494866

		precision	recall	f1-score	support
Not	Potable	0.56	0.60	0.58	234
	Potable	0.61	0.57	0.59	253
	accuracy			0.59	487
	macro avg	0.59	0.59	0.59	487
	weighted avg	0.59	0.59	0.59	487

## Análise

Diminuir o número de instâncias necessárias para que se construa um nó da árvore não constituiu efeito significativo.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

