Water Quality Analysis – Phase 3

Team Member – V.Regin

Performing Exploratory Data Analysis (EDA) through Visualization Techniques to Analyze Water Potability

In this article I will show the Exploratory Data Analysis (EDA) process with visualization techniques to analyze whether a sample of water with a certain content is potable for drinking or not. This process is implemented in the Python programming language.

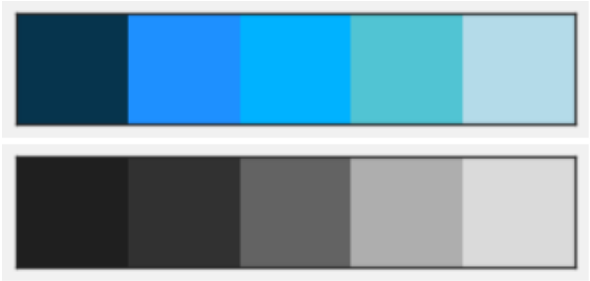
Import Required Libraries

```
import pandas as pd
import numpy as np
from collections import Counter
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import missingno as msno
from warnings import filterwarnings
```

Choose the Desired Color

```
colors = ['#06344d','#00b2ff']
sns.set(palette=colors, font='Serif', style='white', rc={'axes.facecolor':'#f1f1f1', 'figure.facecolor':'#f1f1f1'})
sns.palplot(colors)
```

```
colors_blue = ["#06344d", "#1E90FF", '#00b2ff', '#51C4D3', '#B4DBE9']
colors_dark = ["#1F1F1F", "#313131", '#636363', '#AEAEAE', '#DADADA']
sns.palplot(colors_blue)
sns.palplot(colors_dark)
```



Load Dataset

```
data = pd.read_csv('water_potability.csv')
data.head()
```

From the table it can be seen that the Potability column is a Label (variable to be predicted) and the other columns are Features (variable to predict). The Potability column has two values, namely 0 (not potable) and 1 (potable). So, the Feature columns become parameters to determine whether the water sample is potable (1) or not potable (0). Looking at the dataset, this is a type of **Supervised Learning** task in the form of a **category prediction**.

Check Data Shape and Data Type

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

It can be seen that the data consists of 3276 rows and 10 columns. In addition, the data type of the Potability column (number 9) as Label is integer. We will change the data type of the column to category.

```
data['Potability'] = data['Potability'].astype('category')
data.info()
```

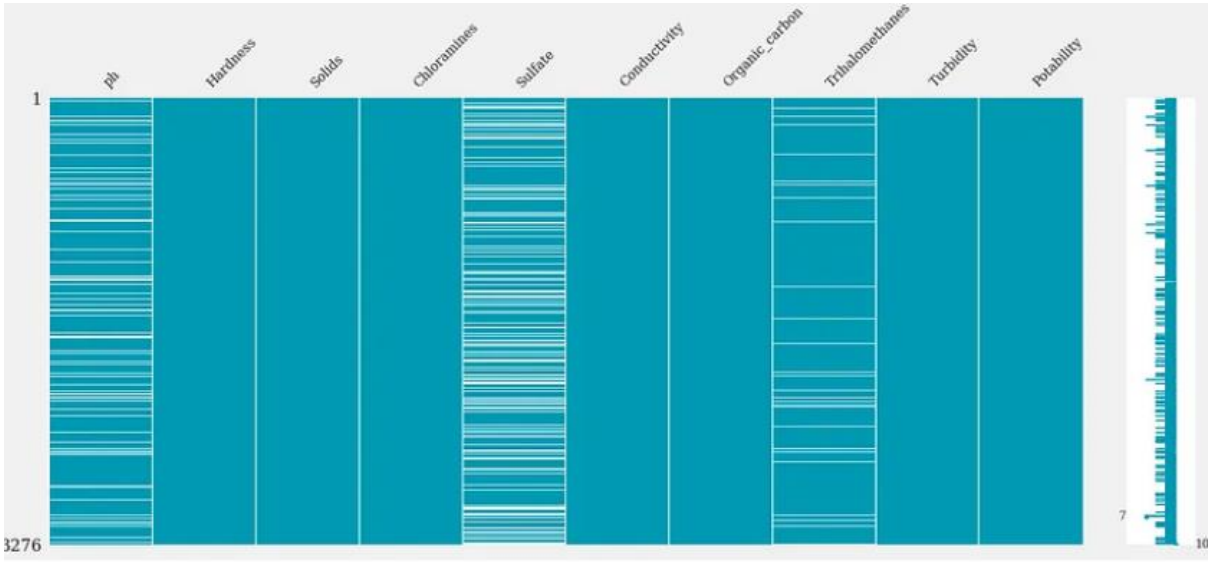
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype   
---  ------           --------------  -----   
 0   ph               2785 non-null   float64 
 1   Hardness         3276 non-null   float64 
 2   Solids           3276 non-null   float64 
 3   Chloramines      3276 non-null   float64 
 4   Sulfate          2495 non-null   float64 
 5   Conductivity     3276 non-null   float64 
 6   Organic_carbon   3276 non-null   float64 
 7   Trihalomethanes  3114 non-null   float64 
 8   Turbidity        3276 non-null   float64 
 9   Potability       3276 non-null   category
dtypes: category(1), float64(9)
memory usage: 233.8 KB
```

Now the data type of the Potability column becomes a category.

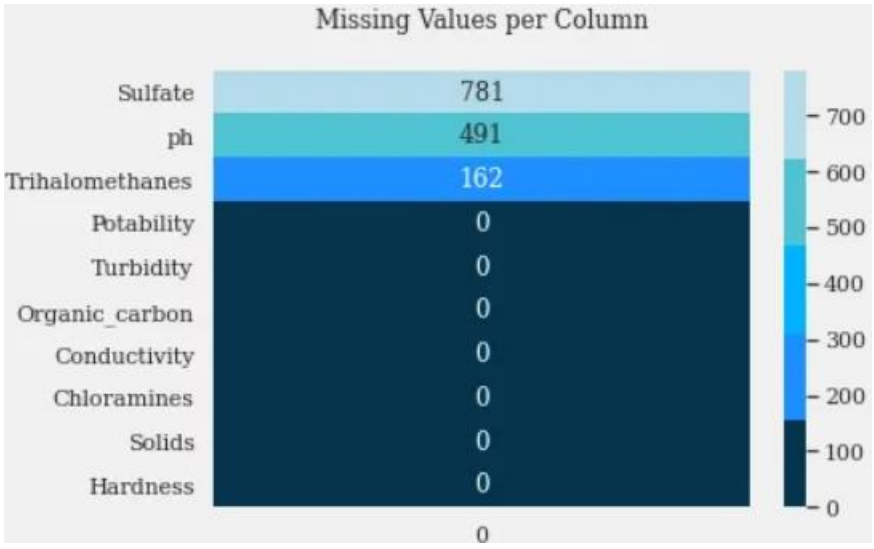
In addition, from the information on the .info() method above, it is shown that there are several columns (pH, Sulfate, and Trihalomethanes) that have fewer rows than the dataset (< 3276). This means that there are empty data values in these columns. Let me show you about this through a heatmap visualization.

Checking Missing Values

```
nan_scratch = msno.matrix(data, color = [0, 0.6, 0.7])
nan_scratch;
```



From the matrix above, the most significant NaN (missing values) is indicated by the Sulfate column. Let's see how many are in heatmap form.

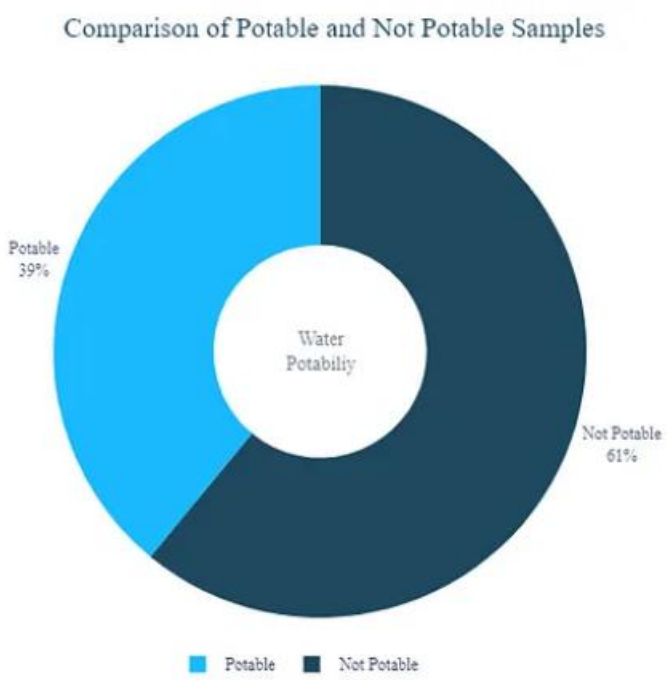


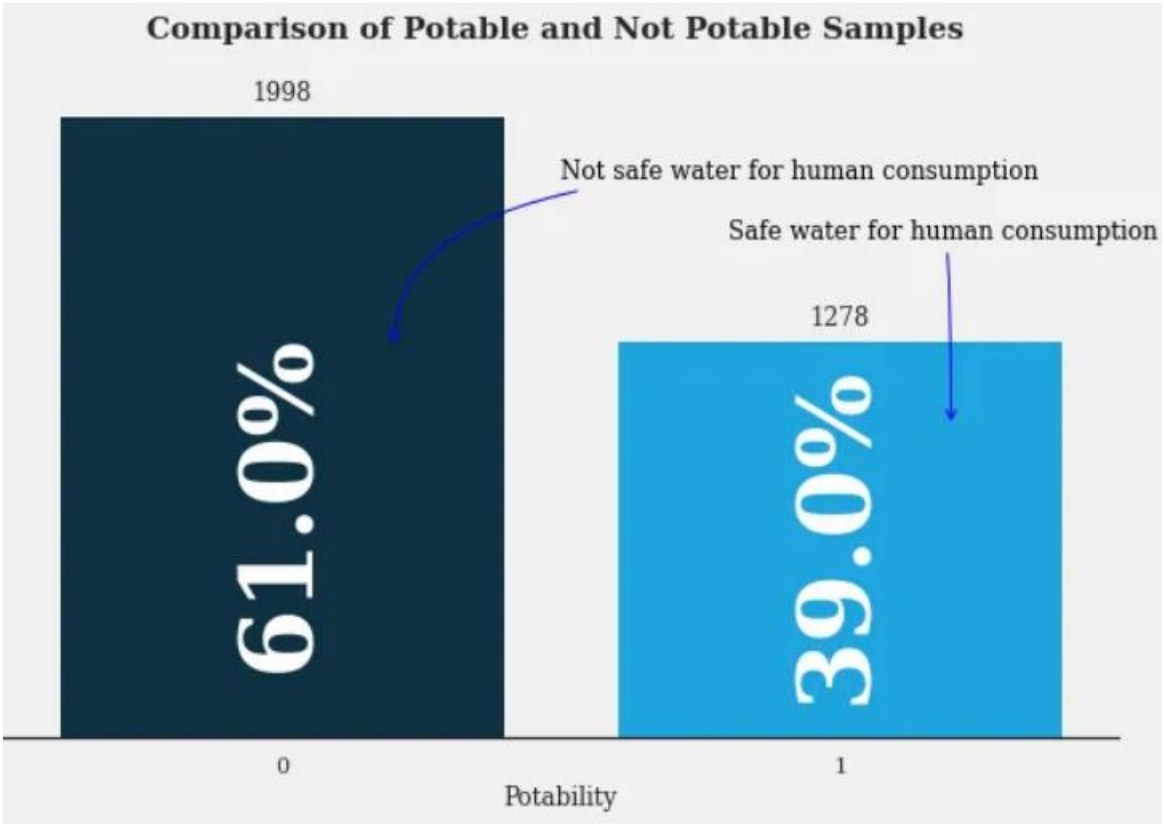
The most important thing here is that the Potability column as a Label does not have a NaN value so that we don't have to bother to impute the Label column which is quite complicated and crucial.

Comparing Water Potability in Dataset

```
df= pd.DataFrame(data['Potability'].value_counts())
fig = px.pie(df,values='Potability',names=['Not Potable','Potable'],hole=0.4,opacity=0.9,
             color_discrete_sequence=[colors[0],colors[1]],labels={'label':'Potability','Potability':'No. Of Samples'})

fig.add_annotation(text='Water<br>Potabiliy', x=0.5,y=0.5,showarrow=False,font_size=14,opacity=0.7,font_family='Gravitas One')

fig.update_layout(font_family='Gravitas One',title=dict(text='Comparison of Potable and Not Potable Samples',x=0.5,y=0.98,
                  font=dict(color=colors_blue[0],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()
```

```
#Visualize the count gender type who visited the Mall
fig = plt.figure(figsize=(10,6))
ax=sns.countplot(data['Potability'], order = data['Potability'].value_counts().index)

#Create annotate
for i in ax.patches:
    ax.text(x=i.get_x()+i.get_width()/2, y=i.get_height()/7, s=f"{np.round(i.get_height()/len(data)*100,0)}%",
            ha='center', size=50, weight='bold', rotation=90, color='white')
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                   (p.get_x() + p.get_width() / 2., p.get_height()),
                   ha = 'center',
                   va = 'center',
                   xytext = (0, 10),
                   textcoords = 'offset points')

plt.title("Comparison of Potable and Not Potable Samples \n", size=15, weight='bold')
plt.annotate(text="Not safe water for human consumption", xytext=(0.5,1790),xy=(0.2,1250),
             arrowprops =dict(arrowstyle="->", color='blue', connectionstyle="angle3,angleA=0,angleB=90"), color='black')
plt.annotate(text="Safe water for human consumption", xytext=(0.8,1600),xy=(1.2,1000),
             arrowprops =dict(arrowstyle="->", color='blue',  connectionstyle="angle3,angleA=0,angleB=90"), color='black')

# Setting Plot
sns.despine(right=True,top = True, left = True)
ax.axes.yaxis.set_visible(False)
plt.setp(ax)
plt.show();
```





It can be seen from the two diagrams that the amount of water which is safe (1) to drink is less than that which is not safe (0).

Checking Correlation of Columns

```
plt.figure(figsize=(16,12))
sns.heatmap(data.corr(), annot=True, cmap = colors_blue) # creating correlational map
plt.title('Correlational Map\n', weight='bold');
```

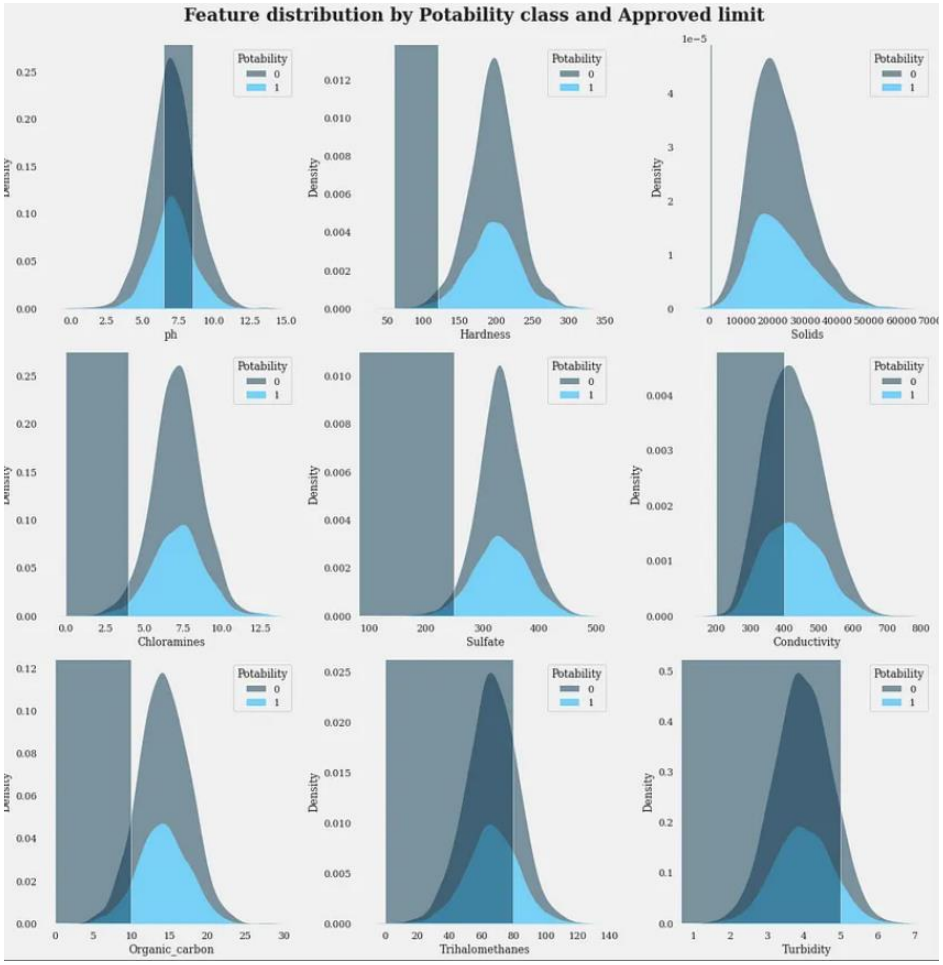
Here, a negative value means a negative correlation, for instance when the value of another column is higher it causes that column to have a lower value. From the heatmap, almost all columns have a very low correlation with other columns except between the pH and Hardness columns. This can be caused by several factors, one of which is the value of the data range in each column that is significantly different. So a **normalization process** is needed.

Checking Data Distribution

The features that affect whether water is fit for consumption have limits that have been standardized by WHO. Let's limit each feature based on information on google.

```
#create approve limit for each features based on data available in Google search
col=data.columns[0:9].to_list()
min_val=[6.5,60,500,0,3,200,0,0,0]
max_val=[8.5,120,1000,4,250,400,10,80,5]
limit=pd.DataFrame(data=[min_val, max_val], columns=col)
```

```
int_cols = data.select_dtypes(exclude=['category']).columns.to_list()
fig, ax= plt.subplots(nrows=3,ncols=3,figsize=(15,15), constrained_layout=True)
plt.suptitle('Feature distribution by Potability class and Approved limit', size=20, weight='bold')
ax=ax.flatten()
for x, i in enumerate(int_cols):
    sns.kdeplot(data=data, x=i, hue='Potability', ax=ax[x], fill=True, multiple='stack', alpha=0.5,
                linewidth=0 )
    l,k = limit.iloc[:,x]
    ax[x].add_patch(Rectangle(xy=(l,0), width=k-l, height=1, alpha=0.5))
    for s in ['left','right','top','bottom']:
        ax[x].spines[s].set_visible(False)
```



Conclusion

There are several conclusions that can be drawn from the overall EDA steps in this article:
- The dataset consists of 3276 rows and 10 columns
- There are 9 columns with float data type and 1 column with category data type
- The Potability column acts as a Label and the other columns as Features
-There are 3 columns with NaN values, namely pH, Sulfate, and Trihalomethanes. Therefore, data imputation needs to be done in these three columns.
- The water quality in the dataset consists of 61% non-potable (not safe for consumption) and 39% potable (safe for consumption)
- Almost all columns have very low correlation with other columns except between Hardness and pH columns (low correlation). Therefore, it is necessary to normalize the data
- Distribution of the water sample that is not potable is more than that of the potable in the Trihalomethanes, Conductivity, and Turbidity columns. In the Solids column almost nothing