

✓ Water Quality Explanatory Data Analysis

What will you learn from this project?

- Bivariate and multivariate data analysis
- Correlation analysis
- Preprocessing: missing value, train-test split and normalization
- Modelling: Decision Tree and Random Forest Classifiers
- Visualize Decision Tree
- Random Forest Hyperparameter Tuning

Introduction

- Access to safe drinking-water is essential to health, a basic human right and a component of effective policy for health protection. This is important as a health and development issue at a national, regional and local level. In some regions, it has been shown that investments in water supply and sanitation can yield a net economic benefit, since the reductions in adverse health effects and health care costs outweigh the costs of undertaking the interventions.
- Drinking water and staying hydrated is associated with a reduced incidence of urinary tract infections (UTIs), lower blood pressure and heart disease. Therefore, drinking water is essential for good heart health.
- Water is the most important nutrient for the body. It has many benefits for your health and helps to protect you from illness and disease. Water is also an essential part of a healthy lifestyle.



Analysis Content

1. [Python Libraries](#)
2. [Data Content](#)
3. [Read and Analyse Data](#)
4. [Dependent Variable Analysis](#)
5. [Correlation Between Features](#)
6. [Distribution of Features](#)
7. [Preprocessing: Missing Value Problem](#)

8. [Preprocessing: Train-Test Split and Normalization](#)
9. [Modelling: Decision Tree and Random Forest Classifiers](#)
10. [Visualize Decision Tree](#)
11. [Random Forest Hyperparameter Tuning](#)
12. [Conclusion](#)

✓ Python Libraries

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-pytho
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import missingno as msno
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserve
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of

# ML
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, RepeatedStratifiedKFold, train_te
from sklearn.metrics import precision_score, confusion_matrix

from sklearn import tree
```

 /kaggle/input/water-potability/water_potability.csv

Data Content

1. **pH value:** PH is an important parameter in evaluating the acid–base balance of water. It is also the indicator of acidic or alkaline condition of water status. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52–6.83 which are in the range of WHO standards.

2. **Hardness:** Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water. Hardness was originally defined as the capacity of water to precipitate soap caused by Calcium and Magnesium.
3. **Solids (Total dissolved solids - TDS):** Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts such as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, sulfates etc. These minerals produced un-wanted taste and diluted color in appearance of water. This is the important parameter for the use of water. The water with high TDS value indicates that water is highly mineralized. Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.
4. **Chloramines:** Chlorine and chloramine are the major disinfectants used in public water systems. Chloramines are most commonly formed when ammonia is added to chlorine to treat drinking water. Chlorine levels up to 4 milligrams per liter (mg/L or 4 parts per million (ppm)) are considered safe in drinking water.
5. **Sulfate:** Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of sulfate is in the chemical industry. Sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L). It ranges from 3 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographic locations.
6. **Conductivity:** Pure water is not a good conductor of electric current rather's a good insulator. Increase in ions concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceeded 400 $\mu\text{S}/\text{cm}$.
7. **Organic_carbon:** Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to US EPA < 2 mg/L as TOC in treated / drinking water, and < 4 mg/Lit in source water which is use for treatment.
8. **Trihalomethanes:** THMs are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is being treated. THM levels up to 80 ppm is considered safe in drinking water.
9. **Turbidity:** The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light emitting properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter. The mean

turbidity value obtained for Wondo Genet Campus (0.98 NTU) is lower than the WHO recommended value of 5.00 NTU.

10. **Potability:** Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.

✓ Read and Analyse Data

```
df = pd.read_csv("/kaggle/input/water-potability/water_potability.csv")
```

```
df.head()
```



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_c
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.3
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.1
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.8
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.4
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.5

```
# describe
df.describe()
```



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620

```
df.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
```

✓ Dependent Variable Analysis

```
d = pd.DataFrame(df["Potability"].value_counts())
fig = px.pie(d, values = "Potability", names = ["Not Potable", "Potable"], hole = 0.35, c
            labels = {"label" : "Potability", "Potability": "Number of Samples"})
fig.update_layout(title = dict(text = "Pie Chart of Potability Feature"))
fig.update_traces(textposition = "outside", textinfo = "percent+label")
fig.show()
```



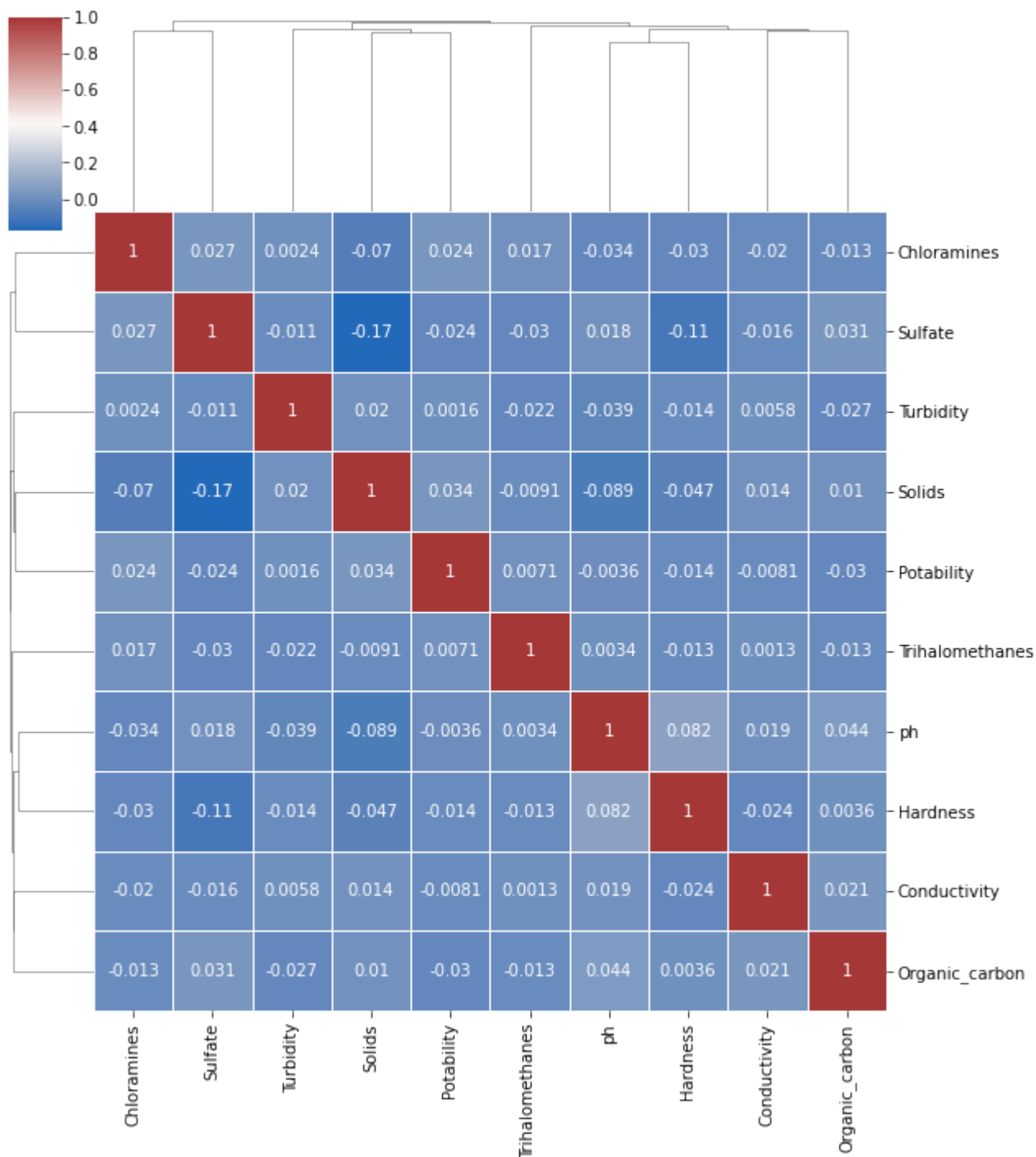
✓ Correlation Between Features

```
df.corr()
```



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
ph	1.000000	0.082096	-0.089288	-0.034350	0.018203	0.018614	0.043503	0.003354	-0.039057	-0.003556
Hardness	0.082096	1.000000	-0.046899	-0.030054	-0.106923	-0.023915	0.003610	-0.013013	-0.014449	-0.013837
Solids	-0.089288	-0.046899	1.000000	-0.070148	-0.171804	0.013831	0.010242	-0.009143	0.019546	0.033743
Chloramines	-0.034350	-0.030054	-0.070148	1.000000	0.027244	-0.020486	-0.012653	0.017084	0.002363	0.023779
Sulfate	0.018203	-0.106923	-0.171804	0.027244	1.000000	-0.016121	0.030831	-0.030274	-0.011187	-0.023577
Conductivity	0.018614	-0.023915	0.013831	-0.020486	-0.016121	1.000000	0.020966	0.001285	0.005798	-0.008128
Organic_carbon	0.043503	0.003610	0.010242	-0.012653	0.030831	0.020966	1.000000	0.001285	0.005798	-0.008128
Trihalomethanes	0.003354	-0.013013	-0.009143	0.017084	-0.030274	0.001285	0.001285	1.000000	0.005798	-0.008128
Turbidity	-0.039057	-0.014449	0.019546	0.002363	-0.011187	0.005798	0.005798	0.005798	1.000000	-0.008128
Potability	-0.003556	-0.013837	0.033743	0.023779	-0.023577	-0.008128	-0.008128	-0.008128	-0.008128	1.000000

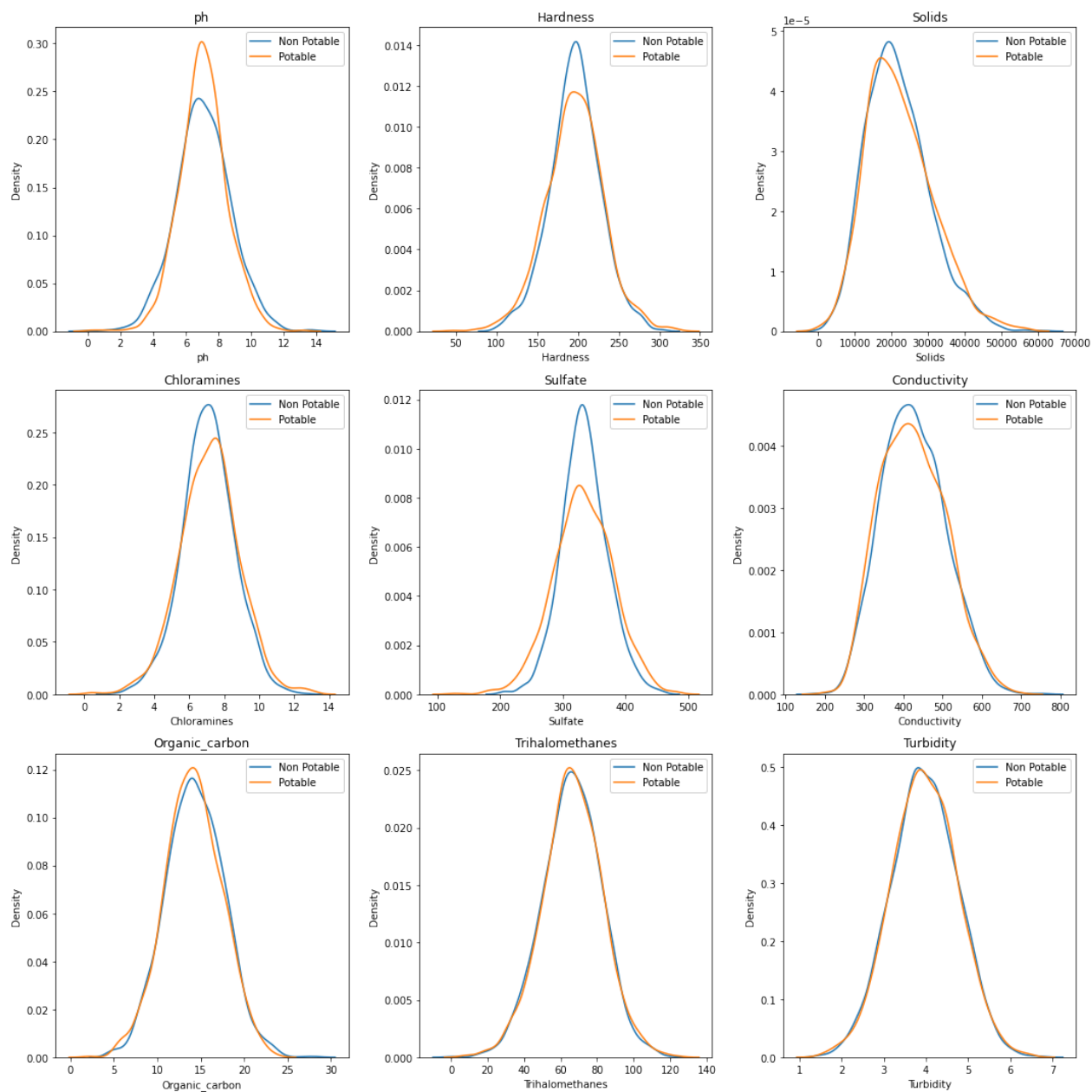
```
sns.clustermap(df.corr(), cmap = "vlag", dendrogram_ratio = (0.1, 0.2), annot = True, lin
plt.show()
```



✓ Distribution of Features

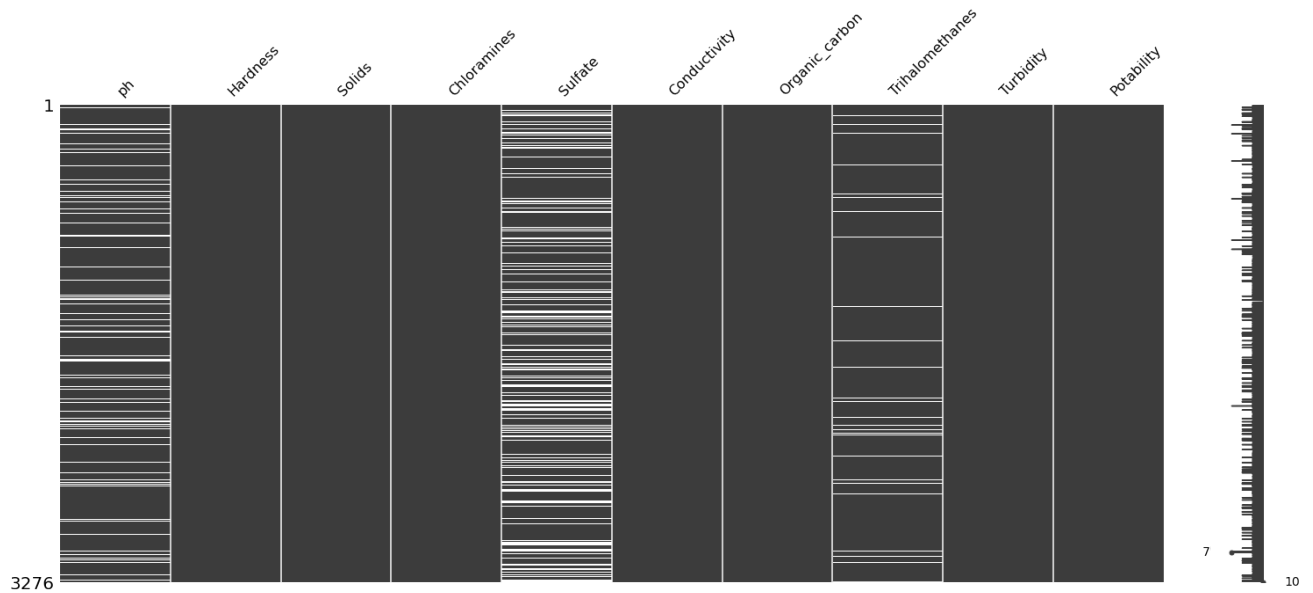
```
non_potable = df.query("Potability == 0")
potable = df.query("Potability == 1")

plt.figure(figsize = (15,15))
for ax, col in enumerate(df.columns[:9]):
    plt.subplot(3,3, ax + 1)
    plt.title(col)
    sns.kdeplot(x = non_potable[col], label = "Non Potable")
    sns.kdeplot(x = potable[col], label = "Potable")
    plt.legend()
plt.tight_layout()
```

✓ Preprocessing: Missing Value Problem

```
msno.matrix(df)
plt.show()
```



```
df.isnull().sum()
```



```
ph          491
Hardness    0
Solids       0
Chloramines  0
Sulfate     781
Conductivity 0
Organic_carbon 0
Trihalomethanes 162
Turbidity   0
Potability  0
dtype: int64
```

```
# handle missing value with average of features
df["ph"].fillna(value = df["ph"].mean(), inplace = True)
df["Sulfate"].fillna(value = df["Sulfate"].mean(), inplace = True)
df["Trihalomethanes"].fillna(value = df["Trihalomethanes"].mean(), inplace = True)

# df.isnull().sum()
```

✓ Preprocessing: Train-Test Split and Normalization

```
X = df.drop("Potability", axis = 1).values
y = df["Potability"].values
```


```
# train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state =
print("X_train",X_train.shape)
print("X_test",X_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
⇒ X_train (2293, 9)
   X_test (983, 9)
   y_train (2293,)
   y_test (983,)
```

```
# min-max normalization
x_train_max = np.max(X_train)
x_train_min = np.min(X_train)
X_train = (X_train - x_train_min)/(x_train_max-x_train_min)
X_test = (X_test - x_train_min)/(x_train_max-x_train_min)
```

✓ Modelling: Decision Tree and Random Forest Classifiers

- Precision Score: The precision is the ratio $tp / (tp + fp)$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

 1_T5mfQqhcn-nB-n7LOiPv6A.png

 1_Ub0nZTXYT8MxLzrz0P7jPA.png

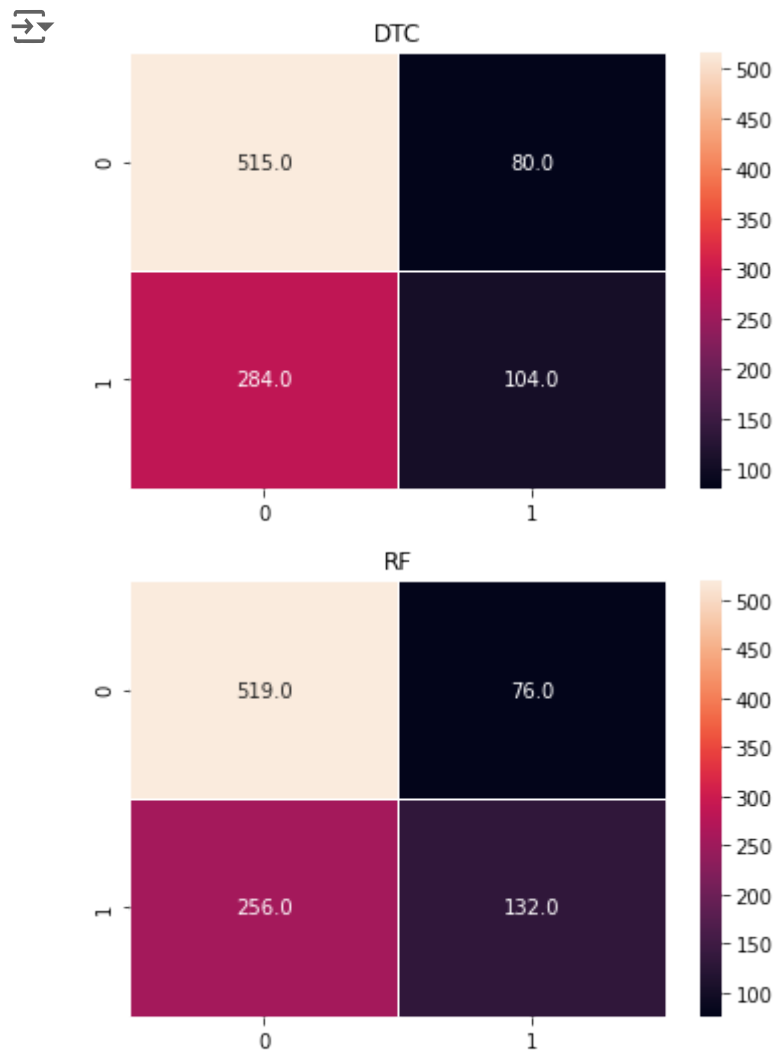
```
models = [("DTC", DecisionTreeClassifier(max_depth = 3)),
          ("RF", RandomForestClassifier())]
```

```
finalResults = []
cmList = []
for name, model in models:
    model.fit(X_train, y_train) # train
    model_result = model.predict(X_test) # prediction
    score = precision_score(y_test, model_result)
    cm = confusion_matrix(y_test, model_result)

    finalResults.append((name, score))
    cmList.append((name, cm))
finalResults
```

```
[('DTC', 0.5652173913043478), ('RF', 0.6346153846153846)]
```

```
for name, i in cmList:
    plt.figure()
    sns.heatmap(i, annot = True, linewidths = 0.8, fmt = ".1f")
    plt.title(name)
    plt.show()
```



```
dt_clf = models[0][1]
dt_clf
```

➡ DecisionTreeClassifier(max_depth=3)

```
plt.figure(figsize = (25,20))
tree.plot_tree(dt_clf,
               feature_names = df.columns.tolist()[:-1],
               class_names = ["0", "1"],
               filled = True,
               precision = 5)
plt.show()
```

