

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_csv("C:\\Users\\Asus\\Desktop\\water_potability.csv")
df
```

```
Out[2]:
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	1.2629758e+00	7.99774743e-01	0	
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	0.0899124	224.236259	19909.541732	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	0.069124	224.236259	19909.541732	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	0.069124	224.236259	19909.541732	0
4	9.062223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	0.069124	224.236259	19909.541732	0
...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419	0.069124	224.236259	19909.541732	0
3272	7.880856	193.553212	17329.802100	8.061362	NaN	392.449580	19.903225	0.069124	224.236259	19909.541732	0
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.039070	0.069124	224.236259	19909.541732	0
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.168946	0.069124	224.236259	19909.541732	0
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.140368	0.069124	224.236259	19909.541732	0

```
In [3]: df.head()
```

```
Out[3]:
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	1.2629758e+00	7.99774743e-01	0	
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	0.0899124	224.236259	19909.541732	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	0.069124	224.236259	19909.541732	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	0.069124	224.236259	19909.541732	0
4	9.062223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	0.069124	224.236259	19909.541732	0

```
In [4]: df.info
```

```
Out[4]: <bound method DataFrame.info of
0      NaN      204.890455      20791.318981      7.300212      368.516441      564.308654      10.379783
1      3.716080      129.422921      18630.057858      6.635246      NaN      592.885359      15.180013
2      8.099124      224.236259      19909.541732      9.275884      NaN      418.606213      16.868637
3      8.316766      214.373394      22018.417441      8.059332      356.886136      363.266516      18.436524
4      9.062223      181.101509      17978.986339      6.546600      310.135738      398.410813      11.558279
...
3271     4.668102      193.681735      47580.991603      7.166639      359.948574      526.424171      13.894419
3272     7.880856      193.553212      17329.802100      8.061362      NaN      392.449580      19.903225
3273     9.419510      175.762646      33155.578218      7.350233      NaN      432.044783      11.039070
3274     5.126763      230.603758      11983.869376      6.303357      NaN      402.883113      11.168946
3275     7.874671      195.102299      17404.177061      7.509306      NaN      327.459760      16.140368
...
Conductivity      Organic_carbon      Trihalomethanes      Turbidity      Potability
0      564.308654      10.379783      86.990978      2.963335
1      592.885359      15.180013      56.329076      4.506056
2      418.606213      16.868637      66.420093      3.055934
3      363.266516      18.436524      100.341674      4.628771
4      398.410813      11.558279      31.997993      4.075075
...
3271     526.424171      13.894419      66.687695      4.435821
3272     392.449580      19.903225      NaN      2.798243
3273     432.044783      11.039070      69.845400      3.298875
3274     402.883113      11.168946      77.488213      4.708658
3275     327.459760      16.140368      78.688446      2.309149
...
[3276 rows x 10 columns]>
```

```
In [5]: df.describe
```

```
Out[5]: <bound method NDFrame.describe of
0      NaN      204.890455      20791.318981      7.300212      368.516441
1      3.716080      129.422921      18630.057858      6.635246      NaN
2      8.099124      224.236259      19909.541732      9.275884      NaN
3      8.316766      214.373394      22018.417441      8.059332      356.886136
4      9.062223      181.101509      17978.986339      6.546600      310.135738
...
3271     4.668102      193.681735      47580.991603      7.166639      359.948574
3272     7.880856      193.553212      17329.802100      8.061362      NaN
3273     9.419510      175.762646      33155.578218      7.350233      NaN
3274     5.126763      230.603758      11983.869376      6.303357      NaN
3275     7.874671      195.102299      17404.177061      7.509306      NaN
...
Conductivity      Organic_carbon      Trihalomethanes      Turbidity      Potability
0      564.308654      10.379783      86.990978      2.963335
1      592.885359      15.180013      56.329076      4.506056
2      418.606213      16.868637      66.420093      3.055934
3      363.266516      18.436524      100.341674      4.628771
4      398.410813      11.558279      31.997993      4.075075
...
3271     526.424171      13.894419      66.687695      4.435821
3272     392.449580      19.903225      NaN      2.798243
3273     432.044783      11.039070      69.845400      3.298875
3274     402.883113      11.168946      77.488213      4.708658
3275     327.459760      16.140368      78.688446      2.309149
...
[3276 rows x 10 columns]>
```

```
In [6]: df.isnull().sum()
```

```
Out[6]: ph      491
Hardness      0
Solids         0
Chloramines    0
Sulfate        0
Conductivity    0
Organic_carbon  0
Trihalomethanes 162
Turbidity       0
Potability      0
dtype: int64
```

```
In [7]: df.corr()
```

```
Out[7]:
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
ph	1.000000	0.082096	-0.089288	-0.034350	0.016203	0.018614	0.04	0.082096	0.082096	0.04
Hardness	0.082096	1.000000	-0.046899	-0.030054	-0.106923	-0.023915	0.00	0.082096	0.082096	0.00
Solids	-0.089288	-0.046899	1.000000	-0.070148	-0.171804	0.013831	-0.01	-0.089288	-0.089288	-0.01
Chloramines	-0.034350	-0.030054	-0.070148	1.000000	0.027244	-0.020486	-0.01	-0.034350	-0.034350	-0.01
Sulfate	0.016203	-0.106923	-0.171804	0.027244	1.000000	-0.016121	0.02	0.016203	0.016203	0.02
Conductivity	0.018614	-0.023915	0.013831	-0.020486	-0.016121	1.000000	0.03	0.018614	0.018614	0.03
Organic_carbon	0.040354	0.003610	0.010242	-0.012653	0.030831	0.020666	1.00	0.040354	0.040354	1.00
Trihalomethanes	0.003354	-0.013013	-0.009143	0.017084	-0.030274	0.001285	-0.01	0.003354	0.003354	-0.01
Turbidity	-0.039057	-0.014449	0.013546	0.002363	0.011187	0.005798	-0.02	-0.039057	-0.039057	-0.02
Potability	-0.003556	-0.013837	0.033743	0.023779	-0.023577	-0.008128	-0.03	-0.003556	-0.003556	-0.03

```
In [8]: df["ph"]=df["ph"].fillna(df["ph"].mean())
df["Sulfate"] = df["Sulfate"].fillna(df["Sulfate"].mean())
df["Trihalomethanes"] = df["Trihalomethanes"].fillna(df["Trihalomethanes"].mean())
```

```
In [9]: df.isna().sum()
```

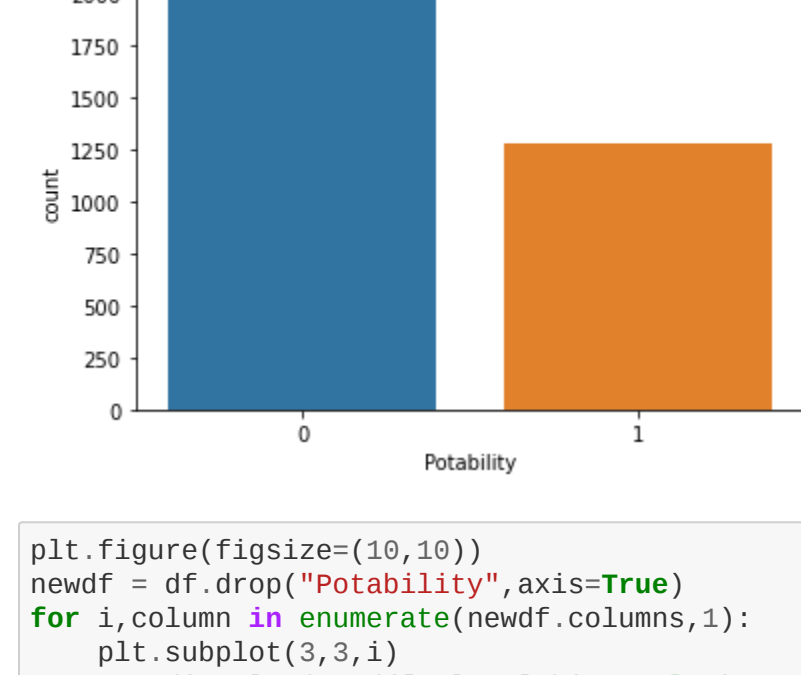
```
Out[9]: ph      0
Hardness      0
Solids         0
Chloramines    0
Sulfate        0
Conductivity    0
Organic_carbon  0
Trihalomethanes 0
Turbidity       0
Potability      0
dtype: int64
```

```
In [10]: df.duplicated().sum()
```

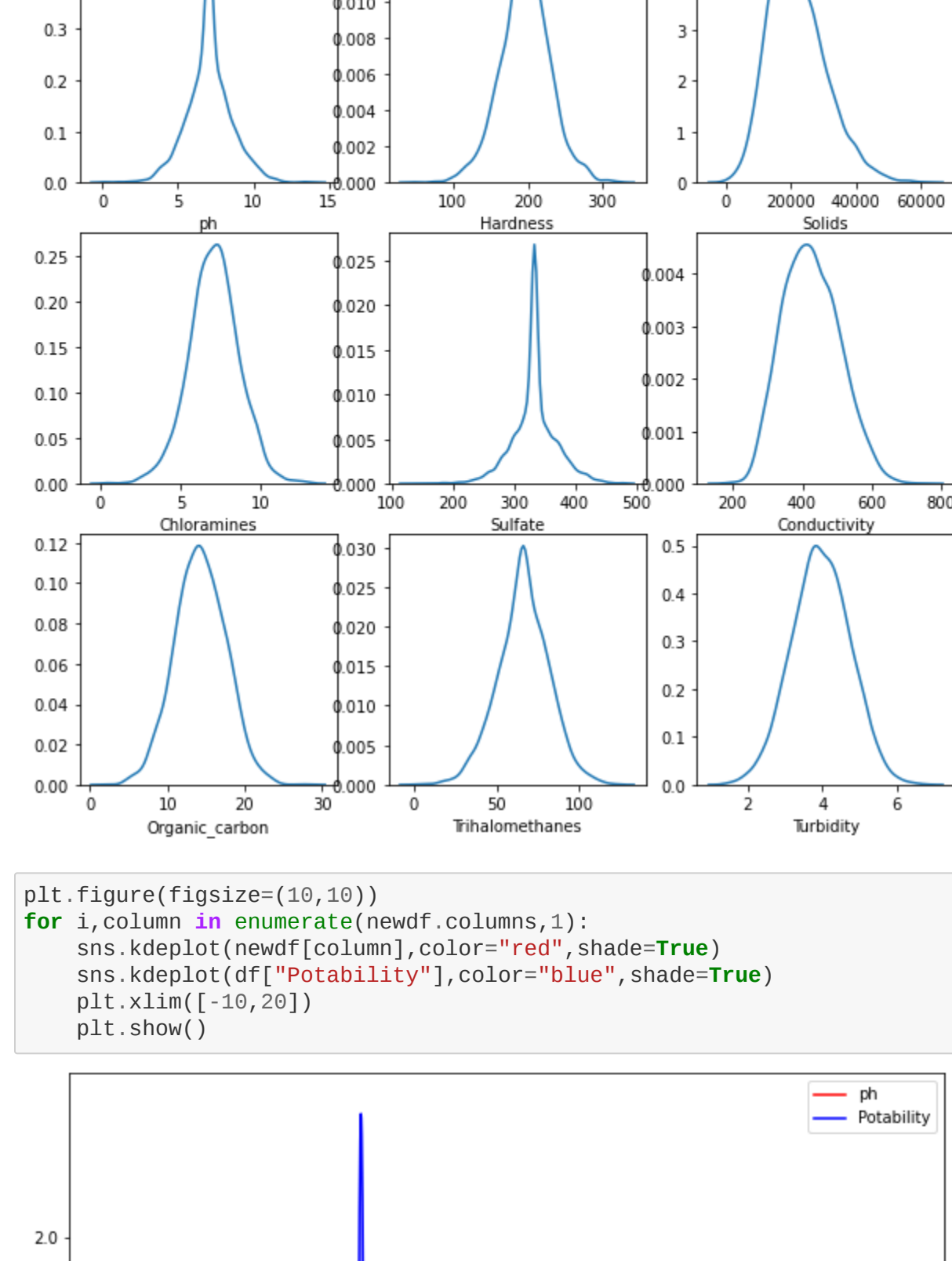
```
Out[10]: 0
```

```
In [12]: sns.countplot(x=df["Potability"],data=df)
```

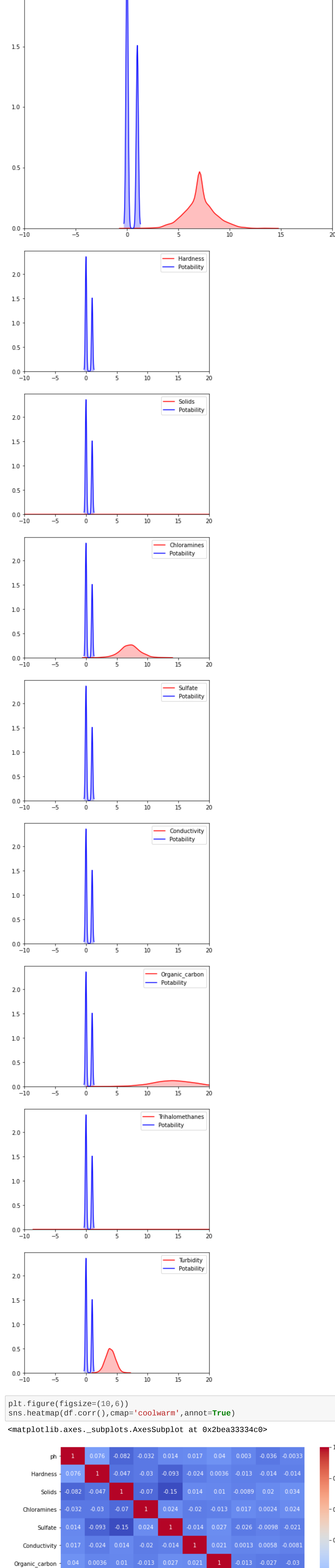
```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x2bea21be06>
```



```
In [14]: plt.figure(figsize=(10,10))
newdf = df.drop("Potability",axis=True)
for i,column in enumerate(newdf.columns,1):
    plt.subplot(3,3,i)
    sns.distplot(newdf[column],hist=False)
```

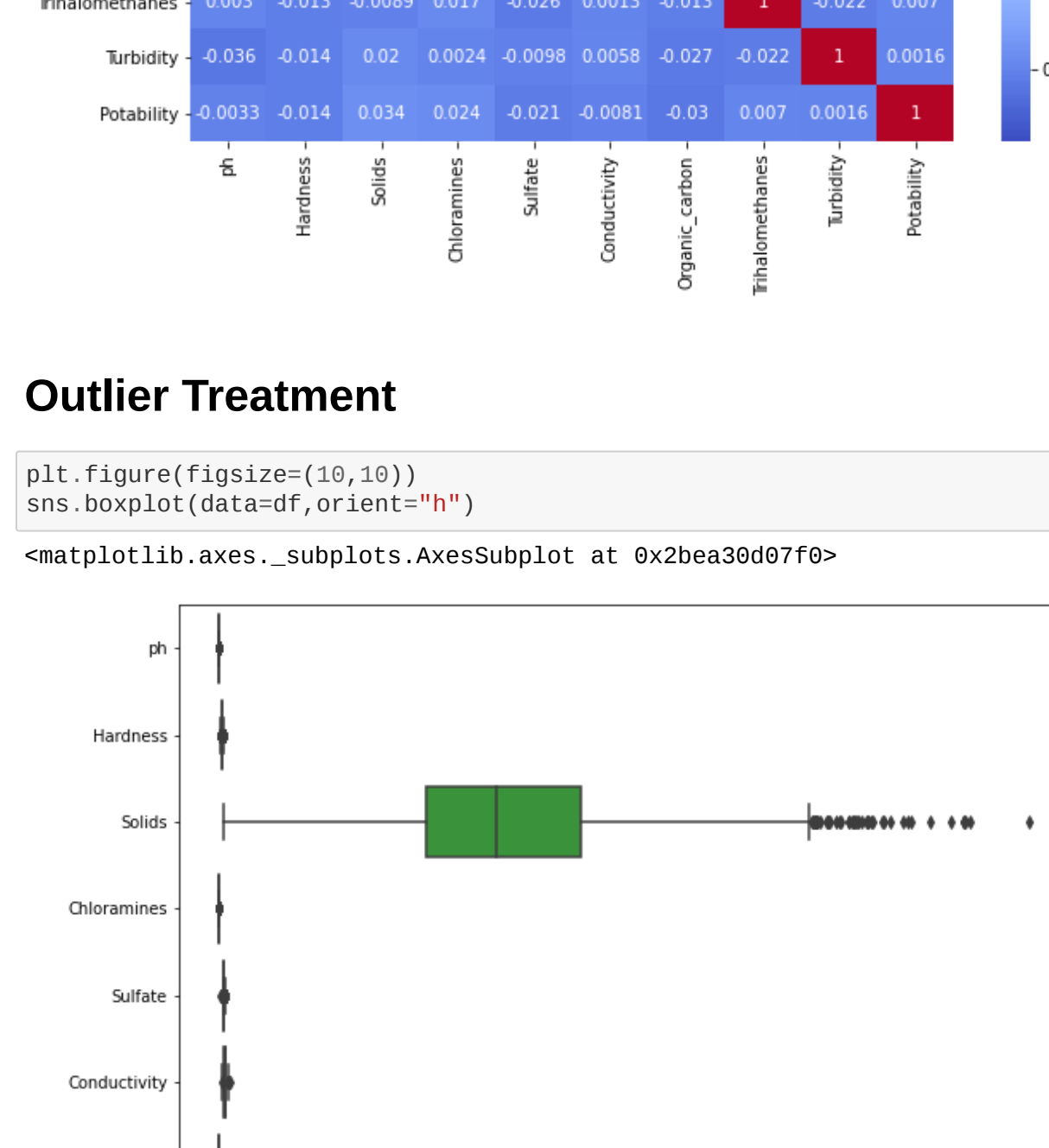


```
In [24]: plt.figure(figsize=(10,10))
for i,column in enumerate(newdf.columns,1):
    sns.kdeplot(newdf[column],color='red',shade=True)
sns.kdeplot(df["Potability"],color='blue',shade=True)
plt.xlim([-10,20])
plt.show()
```



```
In [27]: plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),cmap='coolwarm',annot=True)
```

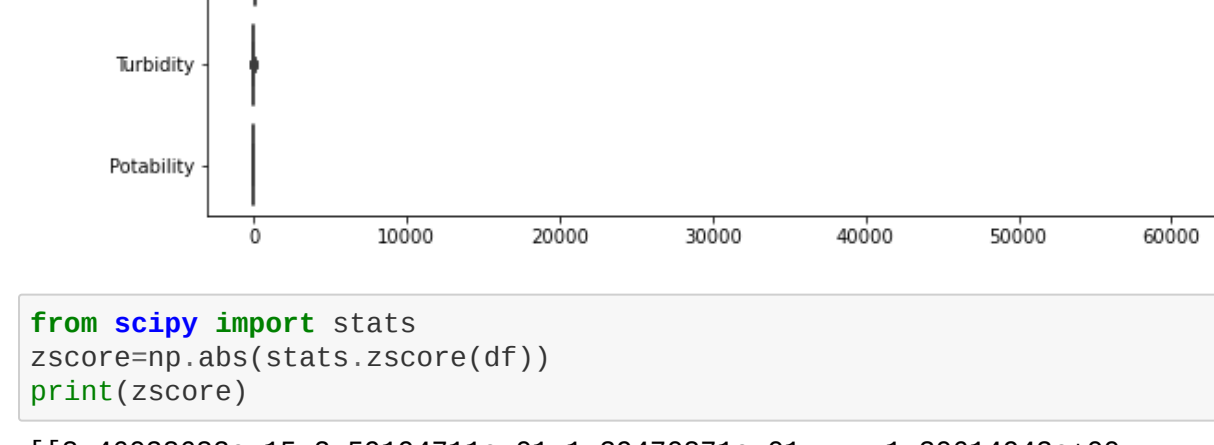
```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x2bea3334c0>
```



Outlier Treatment

```
In [29]: plt.figure(figsize=(10,10))
sns.boxplot(data=df,orient="h")
```

```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x2bea330d7f>
```



```
In [33]: from scipy import stats
zscore=np.abs(stats.zscore(df))
print(zscore)
```

```
[[[ 4.6680683e-15  2.59194711e-01  1.39470871e-01 ... 1.30614943e+00
  1.2629758e+00  7.99774743e-01]
 [ 2.2893338e+00  2.03641367e+00  3.85866550e-01 ... 6.38479983e-01
  6.84217891e-01  7.99774743e-01]
 [ 8.9267789e-01  8.47664833e-01  2.40647337e-01 ... 1.50940884e-03
  1.678654e+00  7.99774743e-01]
 ...
 [ 1.59125368e+00  6.26829230e-01  1.27080989e+00 ... 2.18748247e-01
  5.6060782e-01  1.25035206e+00]
 [ 1.32951593e+00  1.04135450e+00  1.14405809e+00 ... 7.03468439e-01
  9.58797583e-01  1.25035206e+00]
 [ 5.40150905e-01  3.85462310e-02  5.25811937e-01 ... 7.80223466e-01
  2.12445866e+00  1.25035206e+00]]]
```

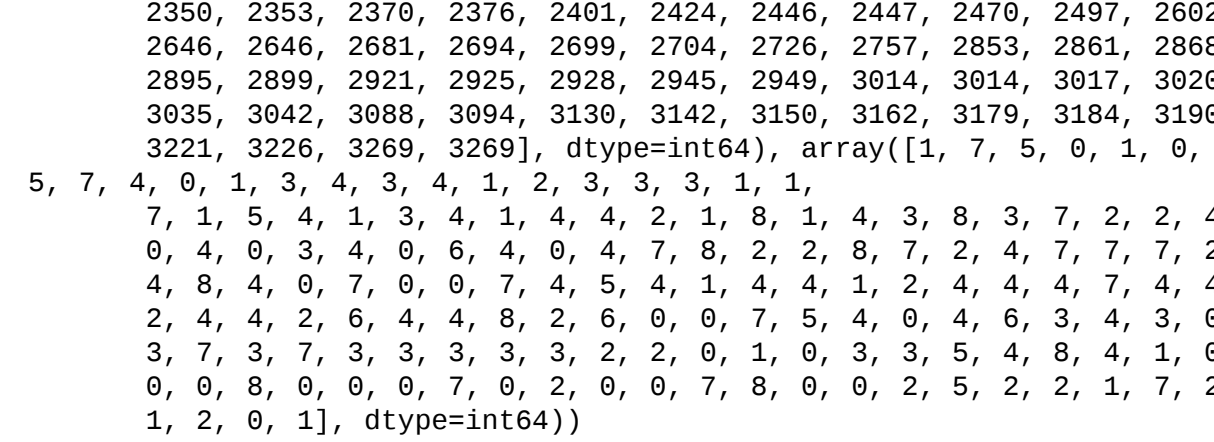
```
In [34]: threshold=3
print(np.where(zscore>3))
```

```
(array([ 37, 61, 66, 80, 88, 104, 148, 245, 253, 263, 26
5,
272, 272, 275, 275, 278, 283, 285, 287, 304, 317, 321,
330, 335, 342, 345, 347, 351, 351, 354, 357, 365, 366,
370, 382, 383, 385, 434, 492, 509, 531, 566, 680,
692, 703, 726, 757, 781, 783, 785, 786, 810, 810, 951,
990, 1031, 1068, 1073, 1075, 1077, 1106, 1123, 1156, 1157, 1186,
1186, 1220, 1292, 1303, 1316, 1343, 1353, 1360, 1366, 1384, 1412,
1490, 1523, 1537, 1542, 1554, 1554, 1563, 1605, 1630, 1642, 1743,
1746, 1766, 1773, 1784, 1792, 1798, 1860, 1892, 1955, 2057, 2075,
2096, 2121, 2134, 2156, 2189, 2204, 2236, 2302, 2318, 2336, 2343,
2350, 2353, 2370, 2376, 2401, 2424, 2446, 2447, 2470, 2497, 2602,
2646, 2646, 2681, 2694, 2699, 2704, 2726, 2757, 2853, 2861, 2868,
2895, 2899, 2921, 2925, 2928, 2945, 2949, 3014, 3014, 3017, 3020,
3035, 3042, 3060, 3094, 3130, 3142, 3150, 3162, 3179, 3184, 3190,
3221, 3226, 3269, 3269], dtype=int64), array([1, 7, 5, 0, 1, 0,
5, 7, 4, 0, 1, 3, 4, 3, 4, 1, 2, 3, 3, 1, 1,
7, 1, 5, 4, 1, 3, 4, 1, 4, 4, 2, 1, 8, 4, 3, 8, 3, 7, 2, 2, 4,
0, 4, 0, 3, 4, 0, 6, 4, 0, 4, 7, 8, 2, 2, 8, 7, 2, 4, 7, 7, 2,
4, 8, 4, 0, 7, 0, 7, 4, 5, 4, 3, 4, 4, 1, 2, 4, 4, 7, 4, 4,
2, 4, 4, 2, 6, 4, 4, 8, 2, 6, 0, 7, 5, 4, 0, 4, 6, 3, 4, 0,
3, 7, 3, 3, 3, 3, 3, 2, 2, 0, 1, 0, 3, 3, 5, 4, 8, 4, 1, 0,
0, 0, 0, 0, 7, 2, 0, 0, 7, 8, 0, 0, 2, 5, 2, 2, 1, 7, 2,
1, 2, 0, ], dtype=int64))
```

```
In [35]: df_clean=df
df_clean = df_clean[(zscore<3).all(axis=1)]
```

```
In [38]: plt.figure(figsize=(10,10))
sns.boxplot(data=df_clean,orient="h")
```

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x2bea3276d0>
```



```
In [40]: df.shape,df_clean.shape
```

```
Out[40]: ((3276, 10), (3128, 10))
```

```
In [41]: x=df.drop(["Potability"],axis=1)
y=df["Potability"]
```

```
In [43]: from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random
states=1)
```

Normalizing and Scaling

```
In [44]: from sklearn.preprocessing import MinMaxScaler
```

```
msscale=MinMaxScaler()
msscale.fit_transform(x_train)
msscale.transform(x_test)
```

```
Out[44]: array([[ 0.4399677,  0.41109394,  0.23731887, ...,  0.38731862,  0.55148795,
  0.38421242],
 [ 0.50577104,  0.67317199,  0.43920245, ...,  0.5819713,  0.41087286,
  0.5359952 ],
 [ 0.60991332,  0.43747643,  0.82024943, ...,  0.37355762,  0.49909904 ,
  0.34989333],
 ...,
 [ 0.42761385,  0.62865848,  0.34328397, ...,  0.36717943,  0.74223794,
  0.6894842 ],
 [ 0.50577104,  0.67317199,  0.2568363, ...,  0.49939759,  0.38501381,
  0.5119923 ],
 [ 0.54389868,  0.55185395,  0.18101919, ...,  0.6025284 ,  0.58387526,
  0.42519848]])
```

Train Models

```
In [45]: from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
In [46]: keys = ['KNN','LogisticRegression','DecisionTree','RandomForest','support
vector','Gaussian']
values = [KNeighborsClassifier(),LogisticRegression(),DecisionTreeClassifier(),RandomForestClassifier(),SVC(),GaussianNB()]
models = dict(zip(keys,values))
print(models)
```

```
['KNN': KNeighborsClassifier(), 'LogisticRegression': LogisticRegression(), 'DecisionTree': DecisionTreeClassifier(), 'RandomForest': RandomForestClassifier(), 'supportvector': SVC(), 'Gaussian': GaussianNB()]
```

```
In [47]: for name,algo in models.items():
model = algo
model.fit(x_train,y_train)
predict = model.predict(x_test)
acc = model.score(x_train,y_train)
print(name,acc)
```

```
KNN:0.650301675389213
LogisticRegression:0.6206106870229008
DecisionTree:1.0
Gaussian:1.0
supportvector:0.6202290076335878
Gaussian:0.634732824427480
```