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

In [2]:
```python
df=pd.read_csv("https://raw.githubusercontent.com/ektanegi25/Water-Sensor-repositor
```

In [79]:
```python
df.head(5)
```

Out[79]:

	Wafers	Sensor-1	Sensor-2	Sensor-3	Sensor-4	Sensor-5	Sensor-6	Sensor-7	Sensor-8	Sensor-9
55	Wafer-856	NaN	2532.45	2191.1333	2197.6570	1.1569	100.0	89.7222	0.1251	1.5762
88	Wafer-889	3221.21	2391.20	2189.9667	1046.6212	0.8662	100.0	102.3622	0.1208	1.4756
26	Wafer-827	2951.85	2525.00	2189.5777	1320.3197	1.3459	100.0	100.7744	0.1234	1.5590
42	Wafer-843	2982.07	2447.06	2199.6334	1242.8420	1.4083	100.0	99.2178	0.1221	1.4542
69	Wafer-870	3058.08	2524.60	2192.3778	1110.5453	0.8147	100.0	99.2922	0.1226	1.4958

5 rows × 592 columns

In [4]:
```python
df.tail(5)
```

Out[4]:

	Unnamed: 0	Sensor-1	Sensor-2	Sensor-3	Sensor-4	Sensor-5	Sensor-6	Sensor-7	Sensor-8	Senso
95	Wafer-896	3013.66	2526.44	2185.2111	1141.6306	0.8447	100.0	100.5978	0.1217	1.533
96	Wafer-897	2982.87	2477.01	2315.2667	2360.1325	1.1259	100.0	90.1144	0.1160	1.469
97	Wafer-898	3084.82	2387.42	2171.5000	1028.4440	0.7899	100.0	101.5122	0.1224	1.360
98	Wafer-899	2955.87	2541.89	NaN	NaN	NaN	NaN	NaN	NaN	1.449
99	Wafer-900	2914.86	2465.11	2210.2778	2120.5760	1.0700	100.0	95.1089	0.1230	1.581

5 rows × 592 columns

In [5]:
```python
df.rename(columns = {'Unnamed: 0':"Wafers"}, inplace = True)
```

In [6]:
```python
df.head(5)
```

Out[6]:

	Wafers	Sensor-1	Sensor-2	Sensor-3	Sensor-4	Sensor-5	Sensor-6	Sensor-7	Sensor-8	Sensor-9	..
0	Wafer-801	2968.33	2476.58	2216.7333	1748.0885	1.1127	100.0	97.5822	0.1242	1.5300	.
1	Wafer-802	2961.04	2506.43	2170.0666	1364.5157	1.5447	100.0	96.7700	0.1230	1.3953	.
2	Wafer-803	3072.03	2500.68	2205.7445	1363.1048	1.0518	100.0	101.8644	0.1220	1.3896	.
3	Wafer-804	3021.83	2419.83	2205.7445	1363.1048	1.0518	100.0	101.8644	0.1220	1.4108	.
4	Wafer-805	3006.95	2435.34	2189.8111	1084.6502	1.1993	100.0	104.8856	0.1234	1.5094	.

5 rows × 592 columns

In [7]:

```
from sklearn.model_selection import train_test_split
```

In [8]:

```
df,df_test=train_test_split(df,test_size=0.2,random_state=42)
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 80 entries, 55 to 51
Columns: 592 entries, Wafers to Good/Bad
dtypes: float64(494), int64(97), object(1)
memory usage: 370.6+ KB
```

In [10]:

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20 entries, 83 to 31
Columns: 592 entries, Wafers to Good/Bad
dtypes: float64(494), int64(97), object(1)
memory usage: 92.7+ KB
```

In [11]:

```
df.describe()
```

Out[11]:

	Sensor-1	Sensor-2	Sensor-3	Sensor-4	Sensor-5	Sensor-6	Sensor-7	Senso
count	79.000000	80.000000	78.000000	78.000000	78.000000	78.0	78.000000	78.0000
mean	3019.048228	2494.058875	2202.758988	1519.467071	1.201382	100.0	96.881160	0.122
std	72.665372	68.166898	31.633772	471.962104	0.365870	0.0	5.520108	0.002
min	2889.670000	2254.990000	2114.666700	978.783200	0.753100	100.0	83.423300	0.116
25%	2975.425000	2452.517500	2189.966700	1111.543600	0.850075	100.0	93.547250	0.120
50%	3004.390000	2502.445000	2200.955600	1308.647900	1.164250	100.0	99.217800	0.122
75%	3065.730000	2532.755000	2212.866700	1997.641600	1.383000	100.0	101.133300	0.123
max	3221.210000	2664.520000	2315.266700	2363.641200	2.207300	100.0	103.091100	0.126

8 rows × 591 columns

In [12]:
```
df['Good/Bad'].isnull().sum()
```

Out[12]:
```
0
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
Wafers         0
Sensor-1       1
Sensor-2       0
Sensor-3       2
Sensor-4       2
              ..
Sensor-587     0
Sensor-588     0
Sensor-589     0
Sensor-590     0
Good/Bad       0
Length: 592, dtype: int64
```

In [14]:
```
df.isnull().sum().sum()
```

Out[14]:
```
1822
```

In [15]:
```
df.isnull().sum().sum()/ (df.shape[0] * (df.shape[1] - 1)) *100
```

Out[15]:
```
3.8536379018612523
```

In [16]:
```
df_test.isnull().sum().sum()
```

Out[16]:
```
484
```

In [17]:
```
df.head()
```

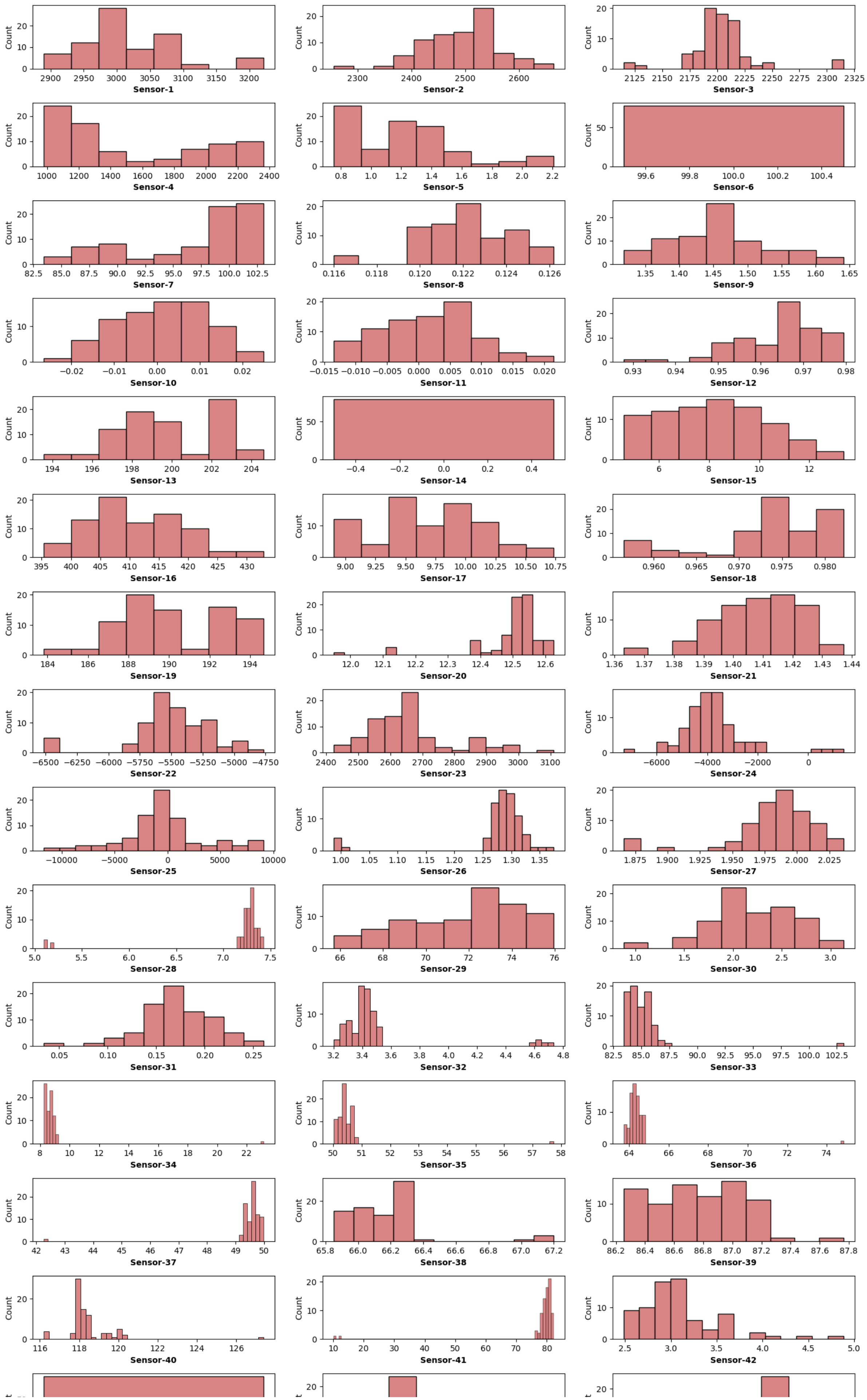
Out[17]:

	Wafers	Sensor-1	Sensor-2	Sensor-3	Sensor-4	Sensor-5	Sensor-6	Sensor-7	Sensor-8	Sensor-9
55	Wafer-856	NaN	2532.45	2191.1333	2197.6570	1.1569	100.0	89.7222	0.1251	1.5762
88	Wafer-889	3221.21	2391.20	2189.9667	1046.6212	0.8662	100.0	102.3622	0.1208	1.4756
26	Wafer-827	2951.85	2525.00	2189.5777	1320.3197	1.3459	100.0	100.7744	0.1234	1.5590
42	Wafer-843	2982.07	2447.06	2199.6334	1242.8420	1.4083	100.0	99.2178	0.1221	1.4542
69	Wafer-870	3058.08	2524.60	2192.3778	1110.5453	0.8147	100.0	99.2922	0.1226	1.4958

5 rows × 592 columns

In [18]:

```
plt.figure(figsize=(15,100))
for i,col in enumerate(df.columns[1:51]):
    plt.subplot(60,3,i+1)
    sns.histplot(x=df[col],color='indianred')
    plt.xlabel(col,weight='bold')
    plt.tight_layout()
```


Count
0 10 20
2900 2950 3000 3050 3100 3150 3200
Sensor-1
Count
0 10 20
2300 2400 2500 2600
Sensor-2
Count
0 10 20
2125 2150 2175 2200 2225 2250 2275 2300 2325
Sensor-3
Count
0 10 20
1000 1200 1400 1600 1800 2000 2200 2400
Sensor-4
Count
0 10 20
0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2
Sensor-5
Count
0 50
99.6 99.8 100.0 100.2 100.4
Sensor-6
Count
0 10 20
82.5 85.0 87.5 90.0 92.5 95.0 97.5 100.0 102.5
Sensor-7
Count
0 10 20
0.116 0.118 0.120 0.122 0.124 0.126
Sensor-8
Count
0 10 20
1.35 1.40 1.45 1.50 1.55 1.60 1.65
Sensor-9
Count
0 10
−0.02 −0.01 0.00 0.01 0.02
Sensor-10
Count
0 10 20
−0.015 −0.010 −0.005 0.000 0.005 0.010 0.015 0.020
Sensor-11
Count
0 10 20
0.93 0.94 0.95 0.96 0.97 0.98
Sensor-12
Count
0 10 20
194 196 198 200 202 204
Sensor-13
Count
0 50
−0.4 −0.2 0.0 0.2 0.4
Sensor-14
Count
0 10
6 8 10 12
Sensor-15
Count
0 10 20
395 400 405 410 415 420 425 430
Sensor-16
Count
0 10
9.00 9.25 9.50 9.75 10.00 10.25 10.50 10.75
Sensor-17
Count
0 10 20
0.960 0.965 0.970 0.975 0.980
Sensor-18
Count
0 10 20
184 186 188 190 192 194
Sensor-19
Count
0 10 20
12.0 12.1 12.2 12.3 12.4 12.5 12.6
Sensor-20
Count
0 10
1.36 1.37 1.38 1.39 1.40 1.41 1.42 1.43 1.44
Sensor-21
Count
0 10 20
−6500 −6250 −6000 −5750 −5500 −5250 −5000 −4750
Sensor-22
Count
0 10 20
2400 2500 2600 2700 2800 2900 3000 3100
Sensor-23
Count
0 10
−6000 −4000 −2000 0
Sensor-24
Count
0 10 20
−10000 −5000 0 5000 10000
Sensor-25
Count
0 10
1.00 1.05 1.10 1.15 1.20 1.25 1.30 1.35
Sensor-26
Count
0 10 20
1.875 1.900 1.925 1.950 1.975 2.000 2.025
Sensor-27
Count
0 10 20
5.0 5.5 6.0 6.5 7.0 7.5
Sensor-28
Count
0 10
66 68 70 72 74 76
Sensor-29
Count
0 10 20
1.0 1.5 2.0 2.5 3.0
Sensor-30
Count
0 10 20
0.05 0.10 0.15 0.20 0.25
Sensor-31
Count
0 10
3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 4.8
Sensor-32
Count
0 10 20
82.5 85.0 87.5 90.0 92.5 95.0 97.5 100.0 102.5
Sensor-33
Count
0 10 20
8 10 12 14 16 18 20 22
Sensor-34
Count
0 10 20
50 51 52 53 54 55 56 57 58
Sensor-35
Count
0 10
64 66 68 70 72 74
Sensor-36
Count
0 10 20
42 43 44 45 46 47 48 49 50
Sensor-37
Count
0 20
65.8 66.0 66.2 66.4 66.6 66.8 67.0 67.2
Sensor-38
Count
0 10
86.2 86.4 86.6 86.8 87.0 87.2 87.4 87.6 87.8
Sensor-39
Count
0 20
116 118 120 122 124 126
Sensor-40
Count
0 10 20
10 20 30 40 50 60 70 80
Sensor-41
Count
0 10
2.5 3.0 3.5 4.0 4.5 5.0
Sensor-42
20
20

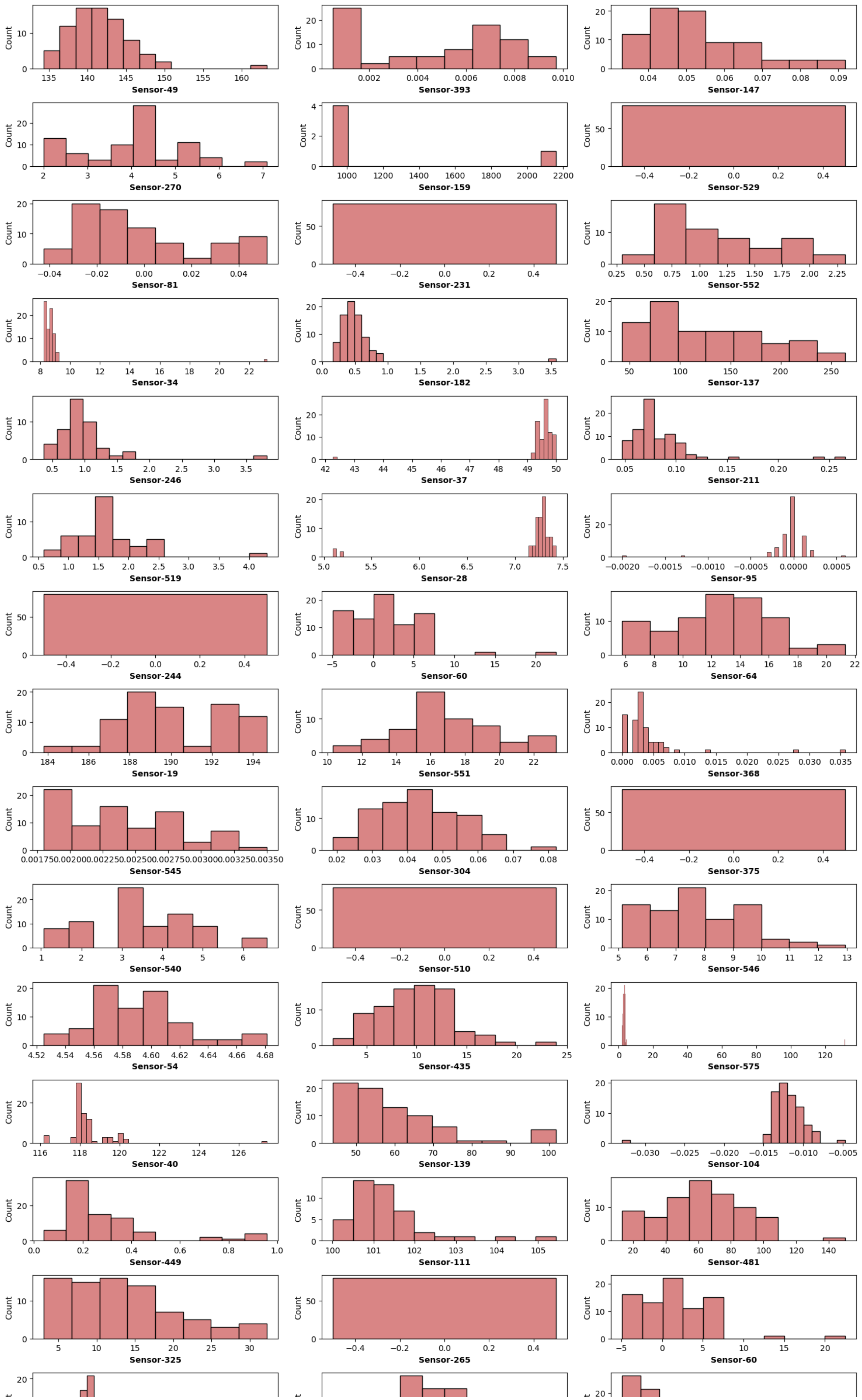
In [22]:

```
random_50_sensors=[]
for i in range(50):
    if i not in random_50_sensors:
        random_50_sensors.append(np.random.randint(1,591))
```

In [20]:

```
#lets have a look to first 50 sensors

plt.figure(figsize=(15,100))
for i,col in enumerate(df.columns[random_50_sensors]):
    plt.subplot(60,3,i+1)
    sns.histplot(x=df[col],color='indianred')
    plt.xlabel(col,weight='bold')
    plt.tight_layout()
```


Count
Sensor-49
135 140 145 150 155 160
Sensor-393
0.002 0.004 0.006 0.008 0.010
Sensor-147
0.04 0.05 0.06 0.07 0.08 0.09
Sensor-270
2 3 4 5 6 7
Sensor-159
1000 1200 1400 1600 1800 2000 2200
Sensor-529
−0.4 −0.2 0.0 0.2 0.4
Sensor-81
−0.04 −0.02 0.00 0.02 0.04
Sensor-231
−0.4 −0.2 0.0 0.2 0.4
Sensor-552
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00 2.25
Sensor-34
8 10 12 14 16 18 20 22
Sensor-182
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Sensor-137
50 100 150 200 250
Sensor-246
0.5 1.0 1.5 2.0 2.5 3.0 3.5
Sensor-37
42 43 44 45 46 47 48 49 50
Sensor-211
0.05 0.10 0.15 0.20 0.25
Sensor-519
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Sensor-28
5.0 5.5 6.0 6.5 7.0 7.5
Sensor-95
−0.0020 −0.0015 −0.0010 −0.0005 0.0000 0.0005
Sensor-244
−0.4 −0.2 0.0 0.2 0.4
Sensor-60
−5 0 5 10 15 20
Sensor-64
6 8 10 12 14 16 18 20 22
Sensor-19
184 186 188 190 192 194
Sensor-551
10 12 14 16 18 20 22
Sensor-368
0.000 0.005 0.010 0.015 0.020 0.025 0.030 0.035
Sensor-545
0.001750.002000.002250.002500.002750.003000.003250.00350
Sensor-304
0.02 0.03 0.04 0.05 0.06 0.07 0.08
Sensor-375
−0.4 −0.2 0.0 0.2 0.4
Sensor-540
1 2 3 4 5 6
Sensor-510
−0.4 −0.2 0.0 0.2 0.4
Sensor-546
5 6 7 8 9 10 11 12 13
Sensor-54
4.52 4.54 4.56 4.58 4.60 4.62 4.64 4.66 4.68
Sensor-435
5 10 15 20 25
Sensor-575
0 20 40 60 80 100 120
Sensor-40
116 118 120 122 124 126
Sensor-139
50 60 70 80 90 100
Sensor-104
−0.030 −0.025 −0.020 −0.015 −0.010 −0.005
Sensor-449
0.0 0.2 0.4 0.6 0.8 1.0
Sensor-111
100 101 102 103 104 105
Sensor-481
20 40 60 80 100 120 140
Sensor-325
5 10 15 20 25 30
Sensor-265
−0.4 −0.2 0.0 0.2 0.4
Sensor-60
−5 0 5 10 15 20

In [23]:
```
def get_cols_zero_std(df:pd.DataFrame):
    cols_to_drop=[]
    num_cols=[i for i in df.columns[1:] if df[i].dtype !="O"]
    for i in df.columns[1:]:
        if df[i].std()==0:
            cols_to_drop.append(i)
    return cols_to_drop
```

In [24]:
```
def get_reduntant_col(df:pd.DataFrame,missing_thresh=.7):
    cols_missing_ratio=df.isnull().sum().div(df.shape[0])
    cols_to_drop = list(cols_missing_ratio[cols_missing_ratio > missing_thresh].ind
    return cols_to_drop
```

In [25]:
```
cols_drop_1=get_cols_zero_std(df=df)
```

In [26]:
```
cols_drop_1
```

```
Out[26]: ['Sensor-6',
          'Sensor-14',
          'Sensor-43',
          'Sensor-50',
          'Sensor-53',
          'Sensor-70',
          'Sensor-75',
          'Sensor-98',
          'Sensor-142',
          'Sensor-150',
          'Sensor-179',
          'Sensor-180',
          'Sensor-187',
          'Sensor-190',
          'Sensor-191',
          'Sensor-192',
          'Sensor-193',
          'Sensor-194',
          'Sensor-195',
          'Sensor-207',
          'Sensor-210',
          'Sensor-227',
          'Sensor-230',
          'Sensor-231',
          'Sensor-232',
          'Sensor-233',
          'Sensor-234',
          'Sensor-235',
          'Sensor-236',
          'Sensor-237',
          'Sensor-238',
          'Sensor-241',
          'Sensor-242',
          'Sensor-243',
          'Sensor-244',
          'Sensor-257',
          'Sensor-258',
          'Sensor-259',
          'Sensor-260',
          'Sensor-261',
          'Sensor-262',
          'Sensor-263',
          'Sensor-264',
          'Sensor-265',
          'Sensor-266',
          'Sensor-267',
          'Sensor-277',
          'Sensor-285',
          'Sensor-314',
          'Sensor-315',
          'Sensor-316',
          'Sensor-323',
          'Sensor-326',
          'Sensor-327',
          'Sensor-328',
          'Sensor-329',
          'Sensor-330',
          'Sensor-331',
          'Sensor-343',
          'Sensor-348',
          'Sensor-365',
          'Sensor-370',
          'Sensor-371',
          'Sensor-372',
          'Sensor-373',
          'Sensor-374',
          'Sensor-375',
          'Sensor-376',
          'Sensor-379',
          'Sensor-380',
          'Sensor-381',
          'Sensor-382',
          'Sensor-395',
          'Sensor-396',
          'Sensor-397',
          'Sensor-398',
          'Sensor-399',
          'Sensor-400',
          'Sensor-401',
          'Sensor-402',
          'Sensor-403',
          'Sensor-404',
          'Sensor-405',
          'Sensor-415',
          'Sensor-423',
          'Sensor-450',
          'Sensor-451',
          'Sensor-452',
          'Sensor-459',
          'Sensor-462',
          'Sensor-463',
          'Sensor-464',
          'Sensor-465',
          'Sensor-466',
          'Sensor-467',
          'Sensor-479',
          'Sensor-482',
          'Sensor-499',
          'Sensor-502',
          'Sensor-503',
          'Sensor-504',
          'Sensor-505',
          'Sensor-506',
          'Sensor-507',
          'Sensor-508',
          'Sensor-509',
          'Sensor-510',
          'Sensor-513',
          'Sensor-514',
          'Sensor-515',
          'Sensor-516',
          'Sensor-529',
          'Sensor-530',
          'Sensor-531',
          'Sensor-532',
          'Sensor-533',
          'Sensor-534',
          'Sensor-535',
          'Sensor-536',
          'Sensor-537',
          'Sensor-538',
          'Sensor-539']
```

```
In [27]: cols_drop_2=get_reduntant_col(df,missing_thresh=.7)
```

```
In [28]: cols_drop_2
```

```
Out[28]: ['Sensor-158', 'Sensor-159', 'Sensor-293', 'Sensor-294']
```

```
In [29]: cols_to_drop=cols_drop_1+cols_drop_2+['Wafers']
```

```
In [30]: len(cols_to_drop)
```

```
Out[30]: 127
```

```
In [31]: df
```

Out[31]:

	Wafers	Sensor-1	Sensor-2	Sensor-3	Sensor-4	Sensor-5	Sensor-6	Sensor-7	Sensor-8	Sensor-9
55	Wafer-856	NaN	2532.45	2191.1333	2197.6570	1.1569	100.0	89.7222	0.1251	1.5762
88	Wafer-889	3221.21	2391.20	2189.9667	1046.6212	0.8662	100.0	102.3622	0.1208	1.4756
26	Wafer-827	2951.85	2525.00	2189.5777	1320.3197	1.3459	100.0	100.7744	0.1234	1.5590
42	Wafer-843	2982.07	2447.06	2199.6334	1242.8420	1.4083	100.0	99.2178	0.1221	1.4542
69	Wafer-870	3058.08	2524.60	2192.3778	1110.5453	0.8147	100.0	99.2922	0.1226	1.4958
...	...	...	...	...	...	...	...	...	...	...
60	Wafer-861	3071.05	2642.15	2200.9889	1054.5240	1.3830	100.0	100.1800	0.1201	1.4532
71	Wafer-872	3043.18	2545.53	2192.3778	1110.5453	0.8147	100.0	99.2922	0.1226	1.3824
14	Wafer-815	3001.26	2519.92	2224.6778	1308.6479	1.3907	100.0	101.1333	0.1208	1.5172
92	Wafer-893	3007.00	2572.62	2213.2111	2070.7147	1.9705	100.0	87.7411	0.1232	1.4446
51	Wafer-852	3078.77	2533.04	2187.4111	1942.3069	1.1864	100.0	88.0911	0.1245	1.4500

80 rows × 592 columns

In [32]:

```
X, y = df.drop(cols_to_drop, axis = 1), df['Good/Bad']
```

In [33]:

```
X.shape
```

Out[33]:

```
(80, 465)
```

In [34]:

```
y.shape
```

Out[34]:

```
(80,)
```

In [35]:

```
y.value_counts()
```

Out[35]:

```
-1    74
 1     6
Name: Good/Bad, dtype: int64
```

In [36]:

```
from sklearn.pipeline import Pipeline
from sklearn.impute import KNNImputer,SimpleImputer
from sklearn.preprocessing import RobustScaler
```

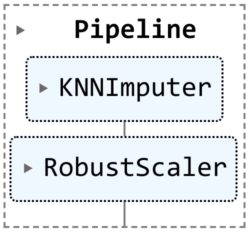
In [37]:

```
imputer=KNNImputer(n_neighbors=3)
prep_process=Pipeline(steps=[('Imputer',imputer),('Scaling',RobustScaler())])
```

In [38]:

```
prep_process
```

Out[38]:



In [39]:

```
X_trans = prep_process.fit_transform(X)
X_trans.shape
```

Out[39]:

```
(80, 465)
```

In [40]:

```
%pip install kneed
```

```
Collecting kneed
  Downloading kneed-0.8.5-py3-none-any.whl (10 kB)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/lib/python3.10/site-pack
ages (from kneed) (1.9.3)
Requirement already satisfied: numpy>=1.14.2 in /opt/conda/lib/python3.10/site-pac
kages (from kneed) (1.23.5)
Installing collected packages: kneed
Successfully installed kneed-0.8.5
Note: you may need to restart the kernel to use updated packages.
```

In [41]:

```
from sklearn.cluster import KMeans
from kneed import KneeLocator
import numpy as np
from typing import Tuple

def cluster_data_instances(X: np.array) -> Tuple[KMeans, np.array]:
    wcss = []  # Within Summation of Squares

    for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
        kmeans.fit(X)
        wcss.append(kmeans.inertia_)

    knee_finder = KneeLocator(
        range(1, 11), wcss, curve='convex', direction='decreasing')
    ideal_clusters = knee_finder.knee

    kmeans = KMeans(n_clusters=ideal_clusters, init='k-means++', random_state=42)
    y_kmeans = kmeans.fit_predict(X)

    return kmeans, np.c_[X, y_kmeans]

# Usage example:
kmeans, X_clus = cluster_data_instances(X_trans)
```

```
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
/opt/conda/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the v
alue of `n_init` explicitly to suppress the warning
  warnings.warn(
```

In [42]:

```
kmeans
```

Out[42]:

```
▾ KMeans
KMeans(n_clusters=3, random_state=42)
```

In [43]:

```
X_clus
```

Out[43]:
```
array([[-0.02781221,  0.37395233, -0.4289214 , ...,  0.75955556,
         0.        ,  1.        ],
       [ 2.50431022, -1.38644649, -0.47986463, ..., -0.52610857,
         0.        ,  1.        ],
       [-0.60204699,  0.28110298, -0.49685153, ..., -0.08775867,
         0.        ,  1.        ],
       ...,
       [-0.03223295,  0.21779093,  1.03590393, ..., -0.17805529,
         0.        ,  2.        ],
       [ 0.03396281,  0.87459106,  0.53517467, ...,  0.28421459,
         0.        ,  1.        ],
       [ 0.86164048,  0.3813055 , -0.59146288, ..., -0.20782888,
         0.        ,  1.        ]])
```

In [44]:
```
np.unique(X_clus[:,-1])
```

Out[44]:
```
array([0., 1., 2.])
```

In [46]:
```
wafer_clus = np.c_[X_clus, y]
wafer_clus
```

Out[46]:
```
array([[-0.02781221,  0.37395233, -0.4289214 , ...,  0.        ,
         1.        , -1.        ],
       [ 2.50431022, -1.38644649, -0.47986463, ...,  0.        ,
         1.        , -1.        ],
       [-0.60204699,  0.28110298, -0.49685153, ...,  0.        ,
         1.        , -1.        ],
       ...,
       [-0.03223295,  0.21779093,  1.03590393, ...,  0.        ,
         2.        , -1.        ],
       [ 0.03396281,  0.87459106,  0.53517467, ...,  0.        ,
         1.        , -1.        ],
       [ 0.86164048,  0.3813055 , -0.59146288, ...,  0.        ,
         1.        , -1.        ]])
```

In [47]:
```
wafer_clus[wafer_clus[:,-2] == 0].shape
```

Out[47]:
```
(1, 467)
```

In [48]:
```
wafer_clus[wafer_clus[:,-2] == 1].shape
```

Out[48]:
```
(62, 467)
```

In [49]:
```
wafer_clus[wafer_clus[:,-2] == 2].shape
```

Out[49]:
```
(17, 467)
```

In [56]:
```
conda install -c conda-forge imbalanced-learn
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 22.11.1
  latest version: 23.9.0

Please update conda by running

    $ conda update -n base -c conda-forge conda

Or to minimize the number of packages updated during conda update use

     conda install conda=23.9.0



## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - imbalanced-learn


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2023.7.22  |       hbcca054_0         146 KB  conda-forge
    certifi-2023.7.22          |     pyhd8ed1ab_0         150 KB  conda-forge
    imbalanced-learn-0.11.0    |     pyhd8ed1ab_0         138 KB  conda-forge
    openssl-3.1.4              |       hd590300_0         2.5 MB  conda-forge
    ------------------------------------------------------------
                                           Total:         2.9 MB

The following NEW packages will be INSTALLED:

  imbalanced-learn   conda-forge/noarch::imbalanced-learn-0.11.0-pyhd8ed1ab_0

The following packages will be UPDATED:

  ca-certificates                     2022.12.7-ha878542_0 --> 2023.7.22-hbcca054
_0
  certifi                           2022.12.7-pyhd8ed1ab_0 --> 2023.7.22-pyhd8ed1
ab_0
  openssl                                 3.0.7-h0b41bf4_1 --> 3.1.4-hd590300_0



Downloading and Extracting Packages
imbalanced-learn-0.1 | 138 KB    |                                       |   0%
ca-certificates-2023 | 146 KB    |                                       |   0%

certifi-2023.7.22    | 150 KB    |                                       |   0%
```

```
openssl-3.1.4        | 2.5 MB    |                                       |   0%

imbalanced-learn-0.1 | 138 KB    | ####2                                 |  12%


openssl-3.1.4        | 2.5 MB    | 2                                     |   1%
ca-certificates-2023 | 146 KB    | ####                                  |  11%

certifi-2023.7.22    | 150 KB    | ##################################### | 100%
ca-certificates-2023 | 146 KB    | ##################################### | 100%


imbalanced-learn-0.1 | 138 KB    | ##################################### | 100%








Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [60]:

```
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
```

In [62]:

```
X, y = X_trans[:,:-1], y
resampler = SMOTE(sampling_strategy= "auto")
X_res, y_res = resampler.fit_resample(X,y)
```

In [63]:

```
print("Before resampling, Shape of training instances: ", np.c_[X, y].shape)
print("After resampling, Shape of training instances: ", np.c_[X_res, y_res].shape)
```

```
Before resampling, Shape of training instances:  (80, 465)
After resampling, Shape of training instances:  (148, 465)
```

In [64]:

```
## Target Cats after Resampling

print(np.unique(y_res))
print(f"Value Counts: \n-1: {len(y_res[y_res == -1])}, 1: {len(y_res[y_res == 1])}"
```

```
[-1  1]
Value Counts:
-1: 74, 1: 74
```

In [65]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=1/3, ra
print(f"train set: {X_train.shape, y_train.shape}")
print(f"test set: {X_test.shape, y_test.shape}")
```

```
train set: ((98, 464), (98,))
test set: ((50, 464), (50,))
```

In [66]:

```
# Prepared training and test sets

X_prep = X_train
y_prep = y_train
X_test_prep = X_test
y_test_prep = y_test

print(X_prep.shape, y_prep.shape)
print(X_test_prep.shape, y_test_prep.shape)
```

```
(98, 464) (98,)
(50, 464) (50,)
```

In [68]:

```
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-2.0.1-py3-none-manylinux2014_x86_64.whl (297.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 297.1/
297.1 MB 6.0 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages (f
rom xgboost) (1.23.5)
Requirement already satisfied: scipy in /opt/conda/lib/python3.10/site-packages (f
rom xgboost) (1.9.3)
Installing collected packages: xgboost
Successfully installed xgboost-2.0.1
```

In [69]:

```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import roc_auc_score


svc_clf = SVC(kernel='linear')
svc_rbf_clf = SVC(kernel='rbf')
random_clf = RandomForestClassifier(random_state=42)
xgb_clf = XGBClassifier()
```

In [70]:

```
## A function to display Scores

def display_scores(scores):
    print("Scores: ", scores)
    print("Mean: ", scores.mean())
    print("Standard Deviation: ", scores.std())
```

In [71]:

```
## SVC Scores

svc_scores = cross_val_score(svc_clf, X_prep, y_prep, scoring='roc_auc', cv=10, ver
display_scores(svc_scores)
```

```
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
Scores:  [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean:  1.0
Standard Deviation:  0.0
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
```

In [72]:
```
## Performance on test set using cross-validation

# Predictions using cross-validation
svc_preds = cross_val_predict(svc_clf, X_test_prep, y_test_prep, cv=5)

# AUC score
svc_auc = roc_auc_score(y_test_prep, svc_preds)
svc_auc
```

Out[72]:
```
0.94
```

In [73]:
```
## SVC rbf Scores

svc_rbf_scores = cross_val_score(svc_rbf_clf, X_prep, y_prep, scoring='roc_auc', cv
display_scores(svc_rbf_scores)
```

```
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
[CV] END .................................................... total time=   0.0s
Scores:  [1.   1.   1.   1.   1.   0.96 1.   1.   1.   1.  ]
Mean:  0.9960000000000001
Standard Deviation:  0.011999999999999976
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.1s finished
```

In [74]:
```
## Performance on test set using cross-validation

# Predictions using cross-validation
svc_rbf_preds = cross_val_predict(svc_rbf_clf, X_test_prep, y_test_prep, cv=5)

# AUC score
```

```
svc_rbf_auc = roc_auc_score(y_test_prep, svc_rbf_preds)
svc_rbf_auc
```

Out[74]:

```
0.6799999999999999
```

In [77]:

```
## Random Forest Scores

random_clf_scores = cross_val_score(random_clf, X_prep, y_prep, scoring='roc_auc',
display_scores(random_clf_scores)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[CV] END .................................................... total time=   0.2s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.2s remaining:    0.0s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
[CV] END .................................................... total time=   0.2s
Scores:  [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Mean:  1.0
Standard Deviation:  0.0
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    2.3s finished
```

In [78]:

```
## Performance on test set using cross-validation

# Predictions using cross-validation
random_clf_preds = cross_val_predict(random_clf, X_test_prep, y_test_prep, cv=5)

# AUC score
random_clf_auc = roc_auc_score(y_test_prep, random_clf_preds)
random_clf_auc
```

Out[78]:

```
1.0
```

In [ ]: