Importing libraries and NOAA reef bleaching dataset

```
%matplotlib
```

```
Using matplotlib backend: Qt5Agg
```

```
from numpy import arange
import numpy
from matplotlib import pyplot as plt
from scipy.stats import norm
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn import model_selection
#from sklearn.metrics import accuracy_score
```

```
plt.rcParams['figure.figsize'] = [16, 7]
```

```
columns = ['Bleaching','Ocean','Year','Depth','Storms','Human Impact','Siltation','Dynamite','Poison','Sewage','Industrial','Comm
df = pd.read_csv(r"C:/Users/Aadya/Downloads/NOAA_reef_check_bleaching_data.csv")
```

Data Exploration

```
df.columns = columns
df.head()
```

	Bleaching	Ocean	Year	Depth	Storms	Human Impact	Siltation	Dynamite	Poison	Sewage	Industrial	Commercial
0	No	Atlantic	2005	4.0	yes	high	often	none	none	high	none	none
1	No	Red Sea	2004	6.0	no	high	occasionally	none	none	low	none	none
2	No	Pacific	1998	3.0	no	low	never	none	none	none	low	none
3	No	Pacific	1998	10.0	no	low	never	none	none	none	low	none
4	No	Atlantic	1997	10.0	no	high	never	none	none	high	moderate	none

```
#Information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9111 entries, 0 to 9110
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Bleaching     9111 non-null   object 
 1   Ocean         9111 non-null   object 
 2   Year          9111 non-null   int64  
 3   Depth         9111 non-null   float64
 4   Storms        9111 non-null   object 
 5   Human Impact  9111 non-null   object 
 6   Siltation     9111 non-null   object 
 7   Dynamite      9111 non-null   object 
 8   Poison        9111 non-null   object 
 9   Sewage        9111 non-null   object 
 10  Industrial    9111 non-null   object 
 11  Commercial    9111 non-null   object 
dtypes: float64(1), int64(1), object(10)
memory usage: 854.3+ KB
```

```
#Data types of the dataset columns
df.dtypes
```

```
Bleaching        object
Ocean            object
Year              int64
Depth           float64
Storms           object
Human Impact     object
Siltation        object
Dynamite         object
Poison           object
Sewage           object
Industrial       object
Commercial       object
dtype: object
```

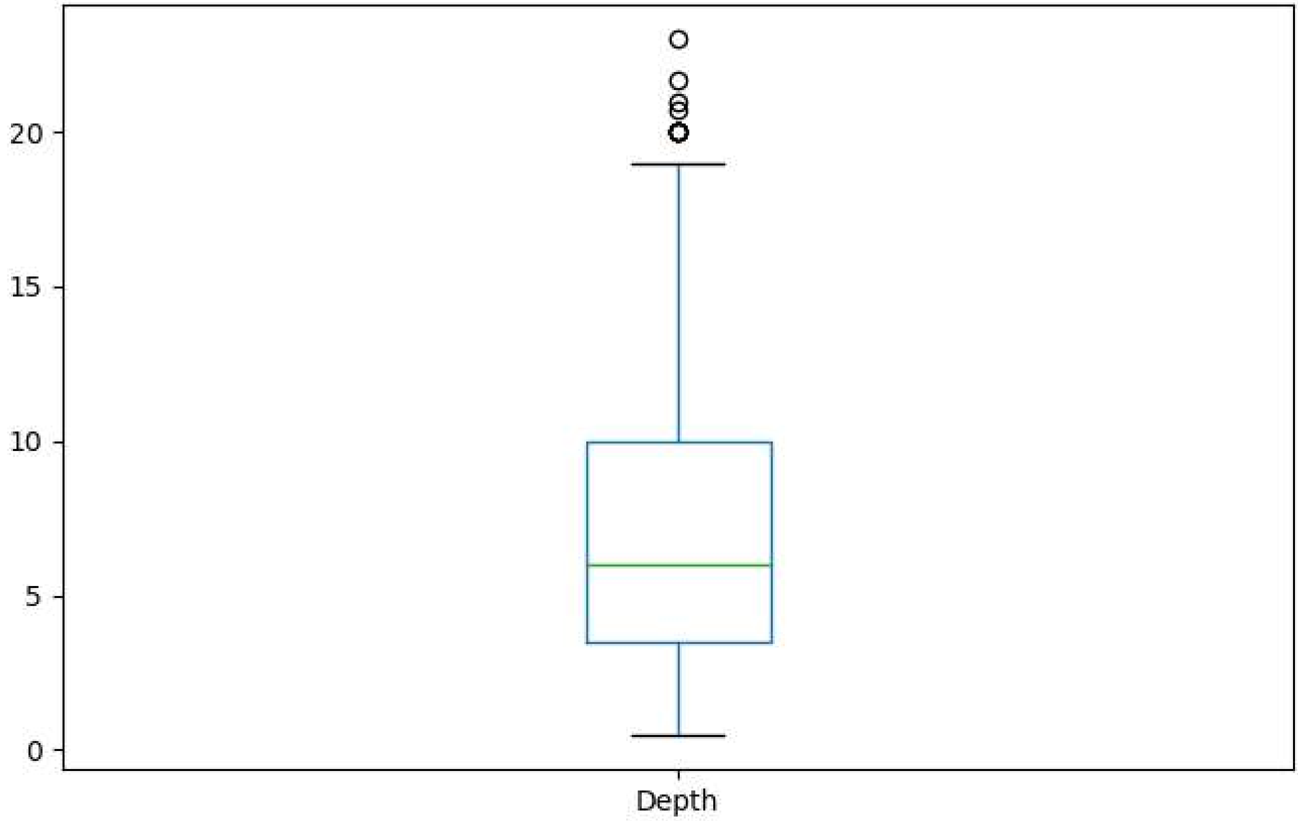
```
#Memory used by each column in the dataset
df.memory_usage()
```

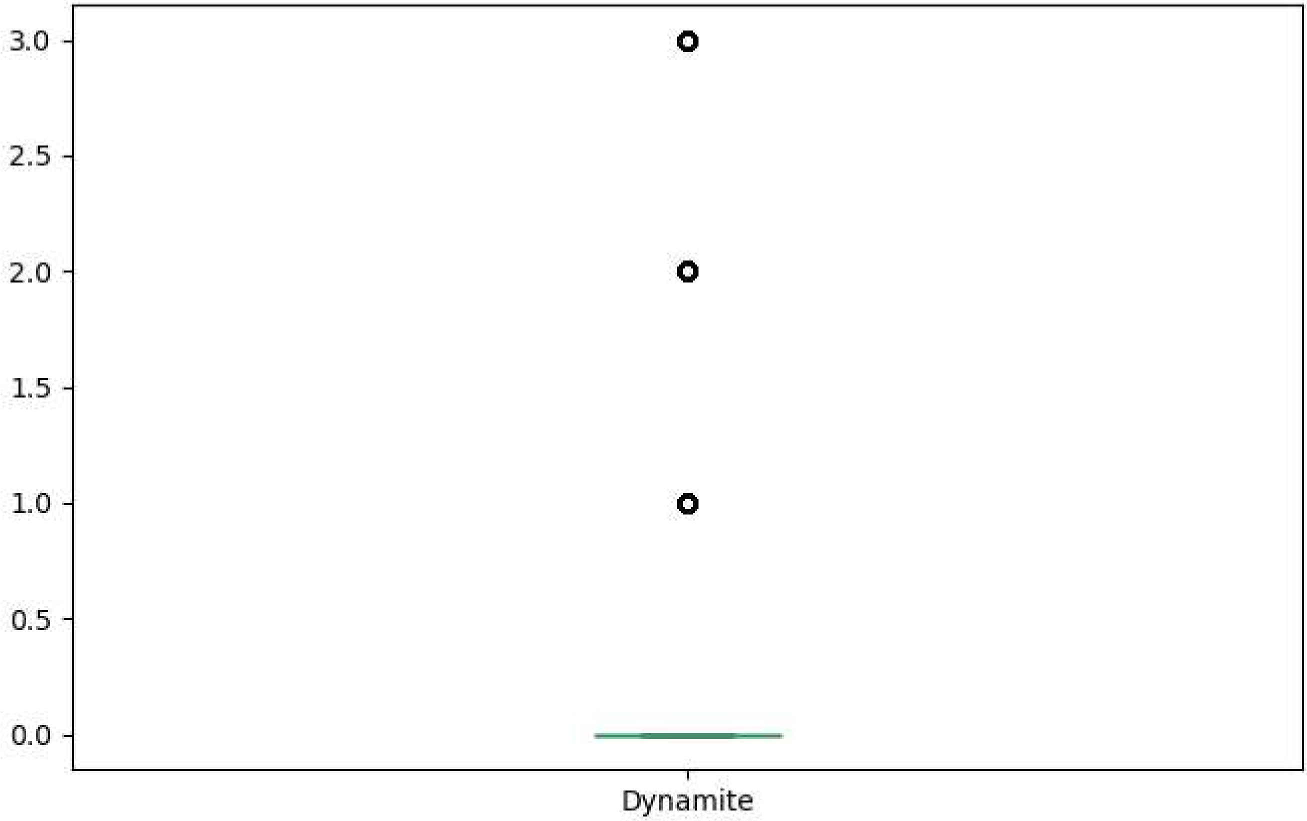
```
Index             128
Bleaching       72888
Ocean           72888
Year            72888
Depth           72888
Storms          72888
Human Impact    72888
Siltation       72888
Dynamite        72888
Poison          72888
Sewage          72888
Industrial      72888
Commercial      72888
dtype: int64
```

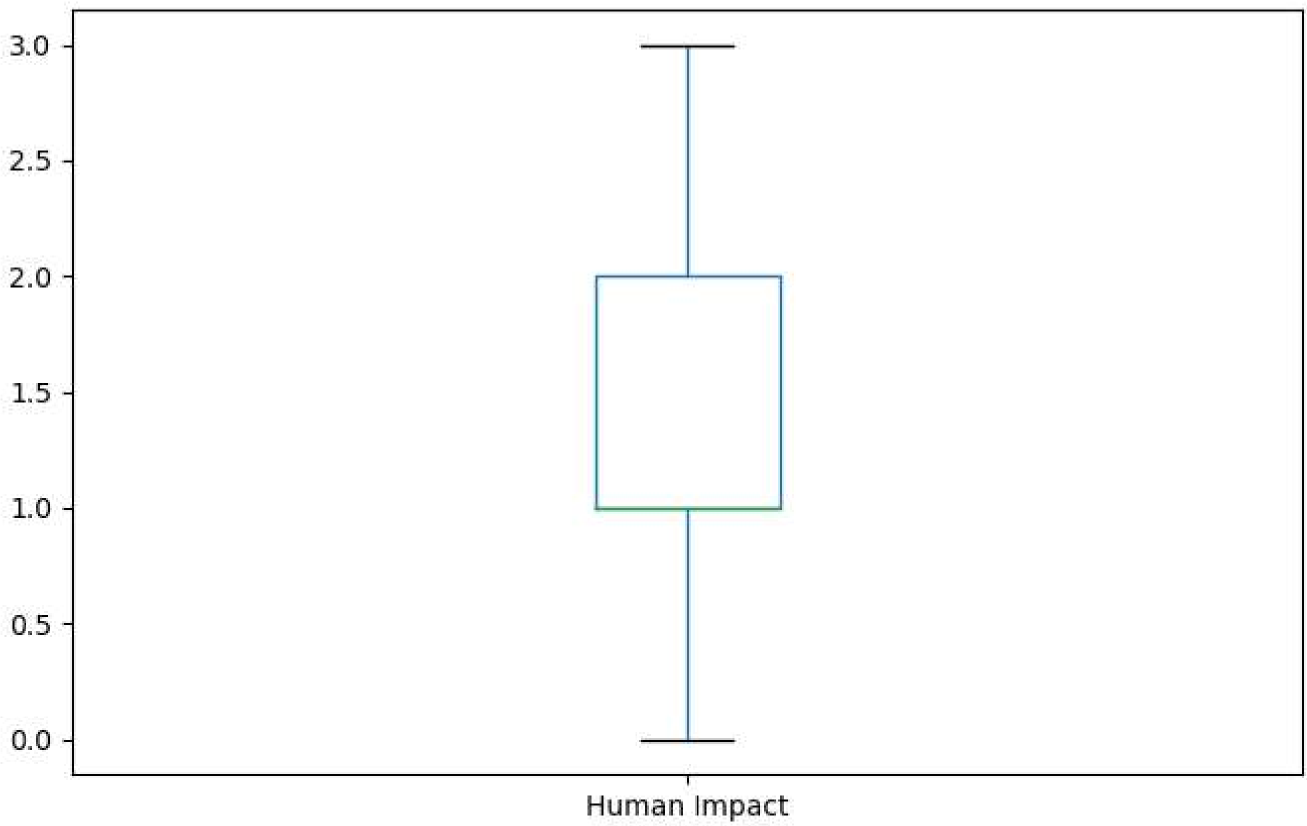
```
#Total memory used by the dataset
df.memory_usage().sum()
```

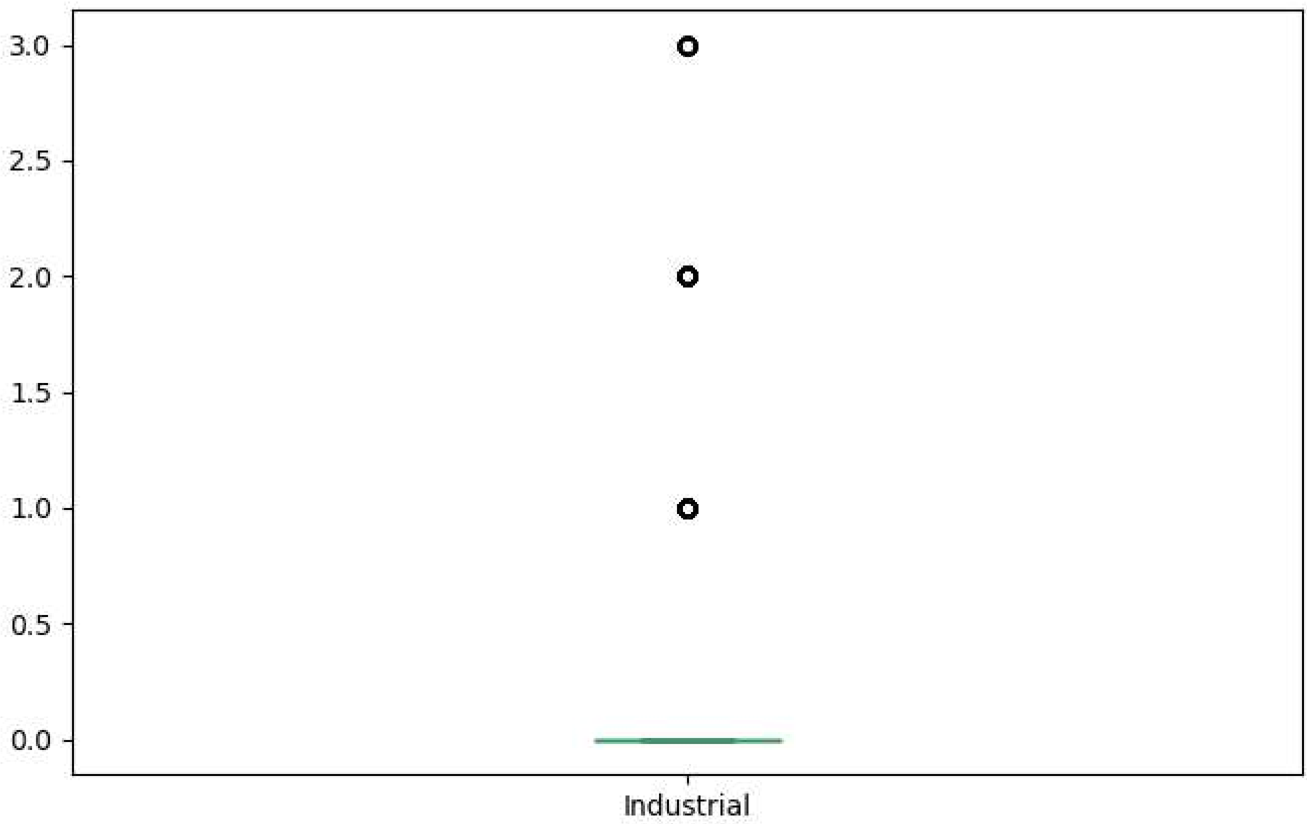
```
874784
```

```
#Barplot
df_avg_depth = df.groupby('Year')['Human Impact'].mean()
df_avg_depth[:].plot.bar(color='orange');
```

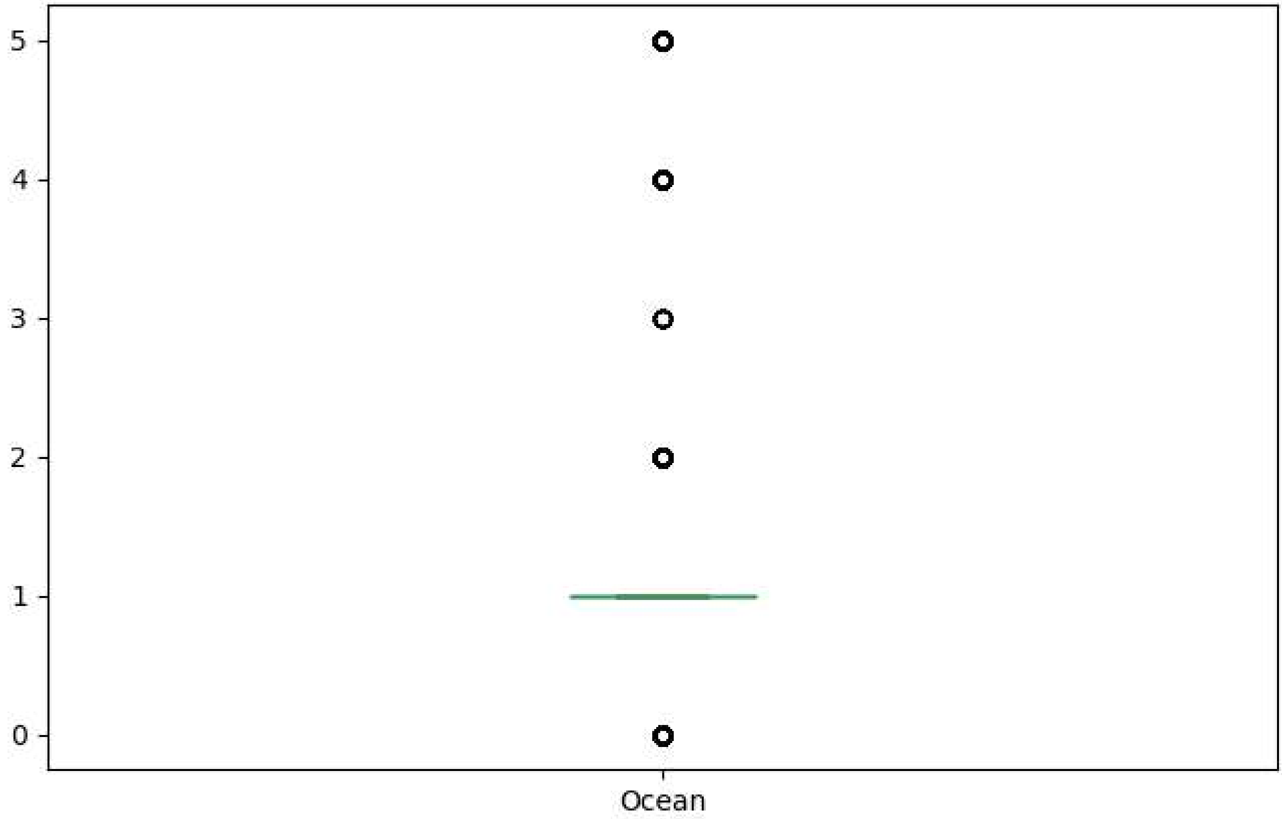

20
15
10
5
0
Depth

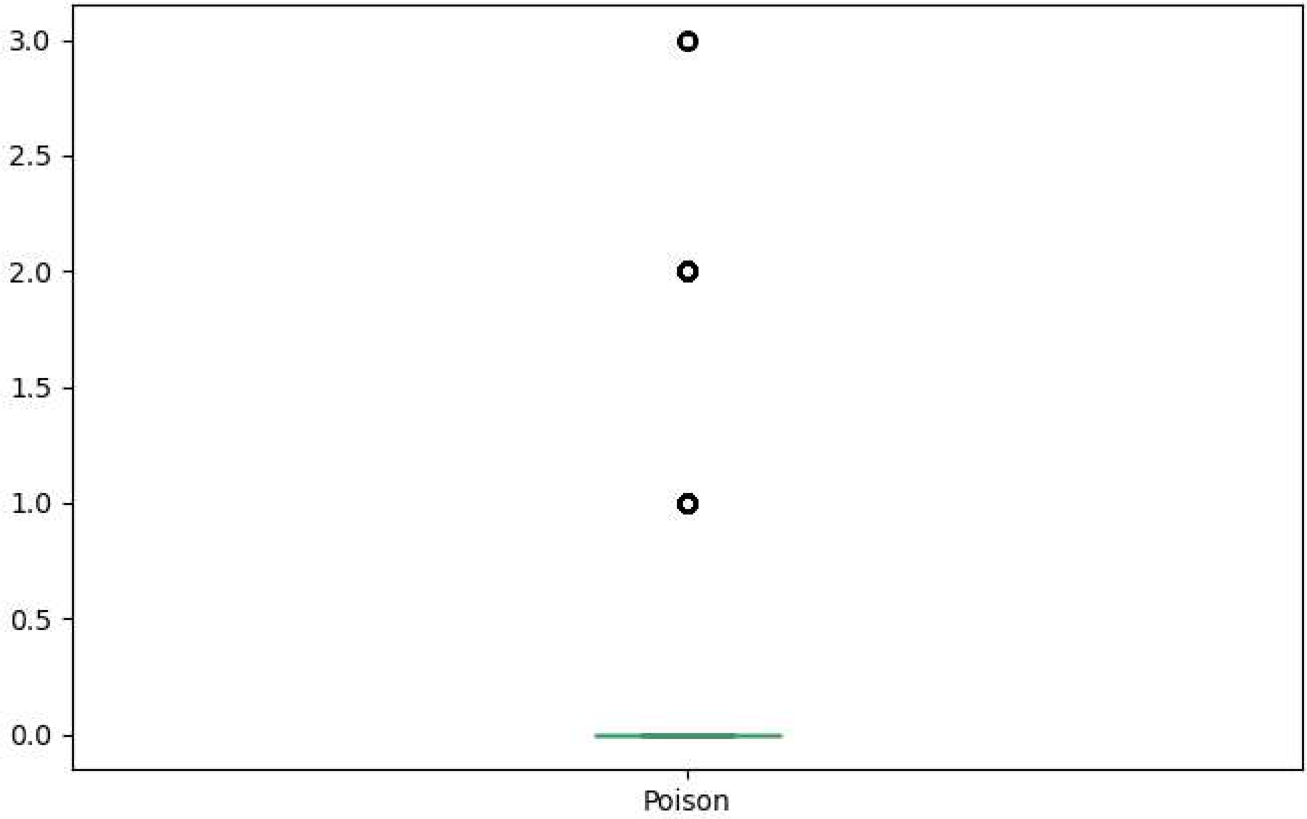

3.0
2.5
2.0
1.5
1.0
0.5
0.0
Dynamite


3.0
2.5
2.0
1.5
1.0
0.5
0.0
Human Impact

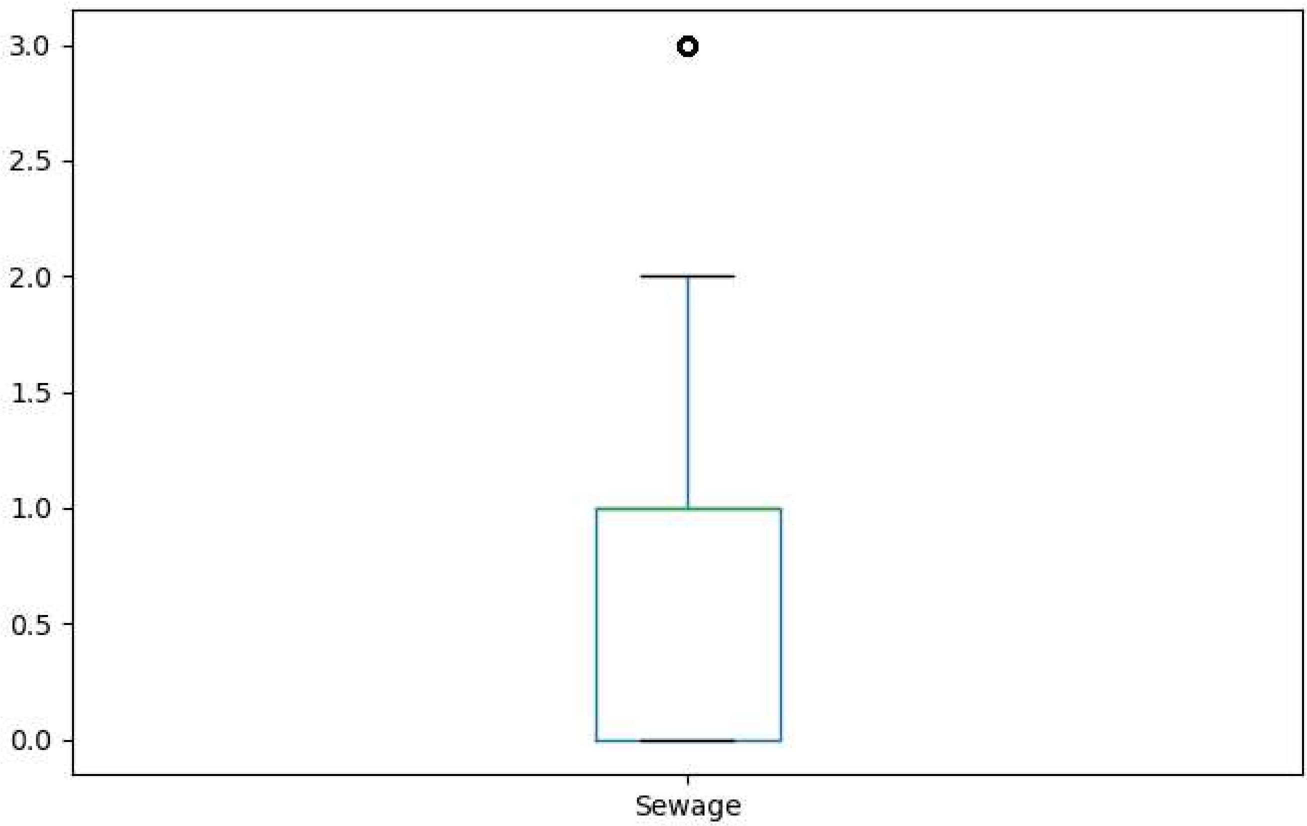


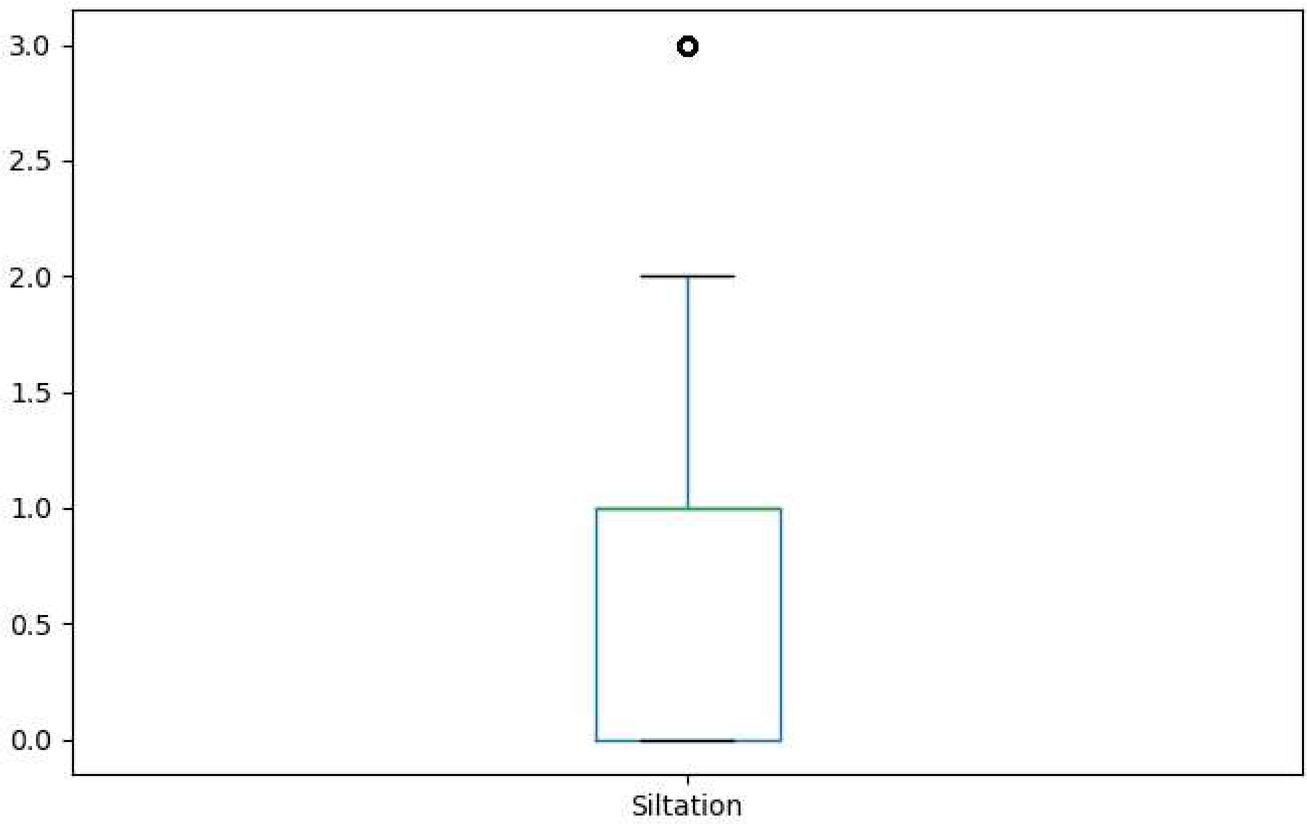
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Industrial

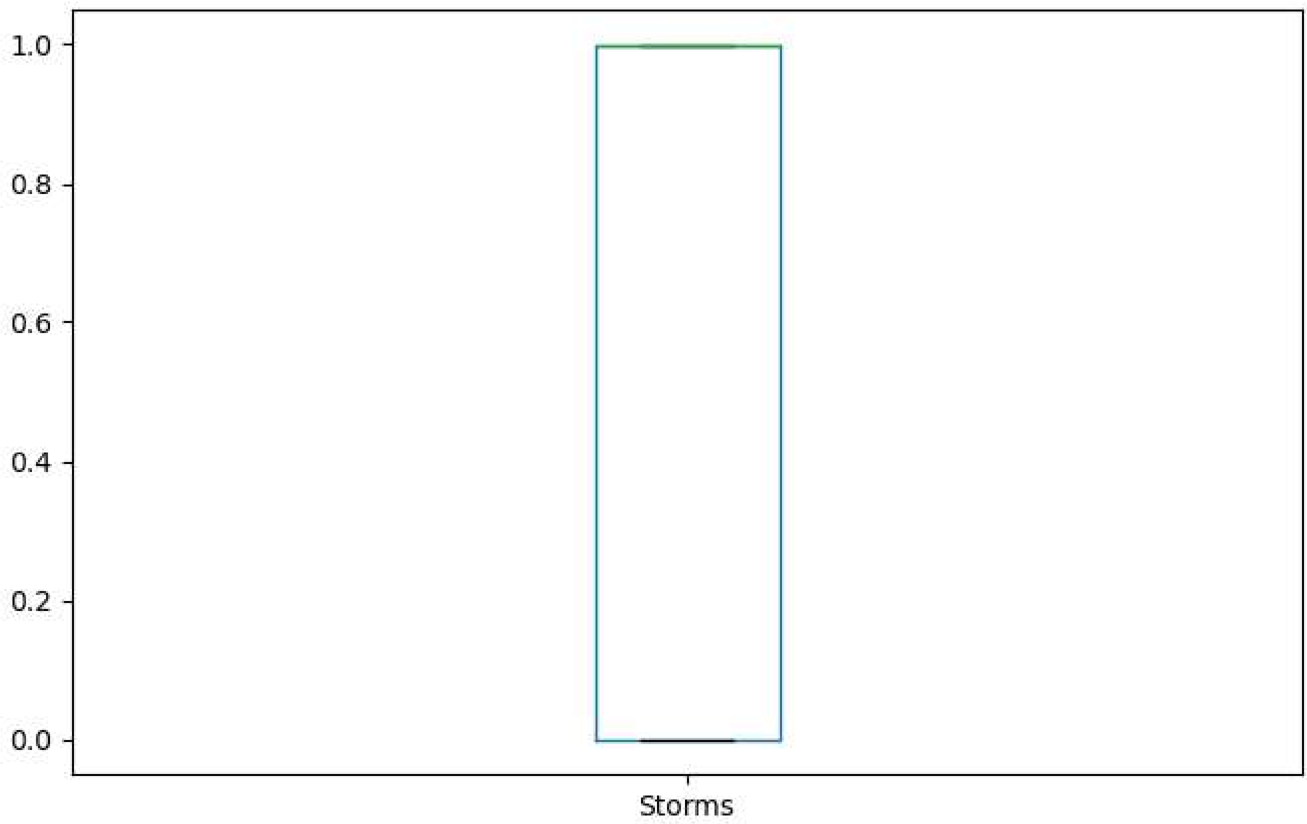

5
4
3
2
1
0
Ocean



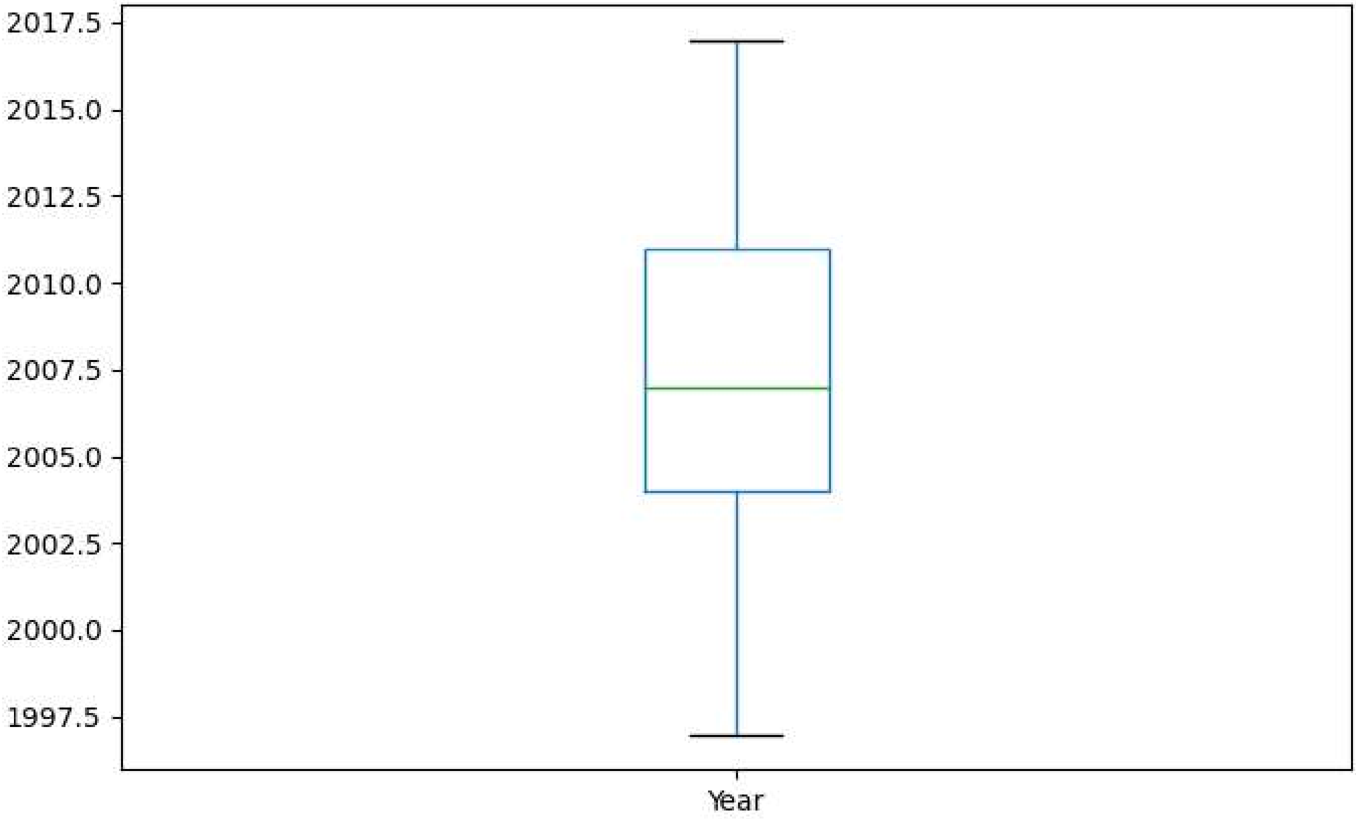
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Poison

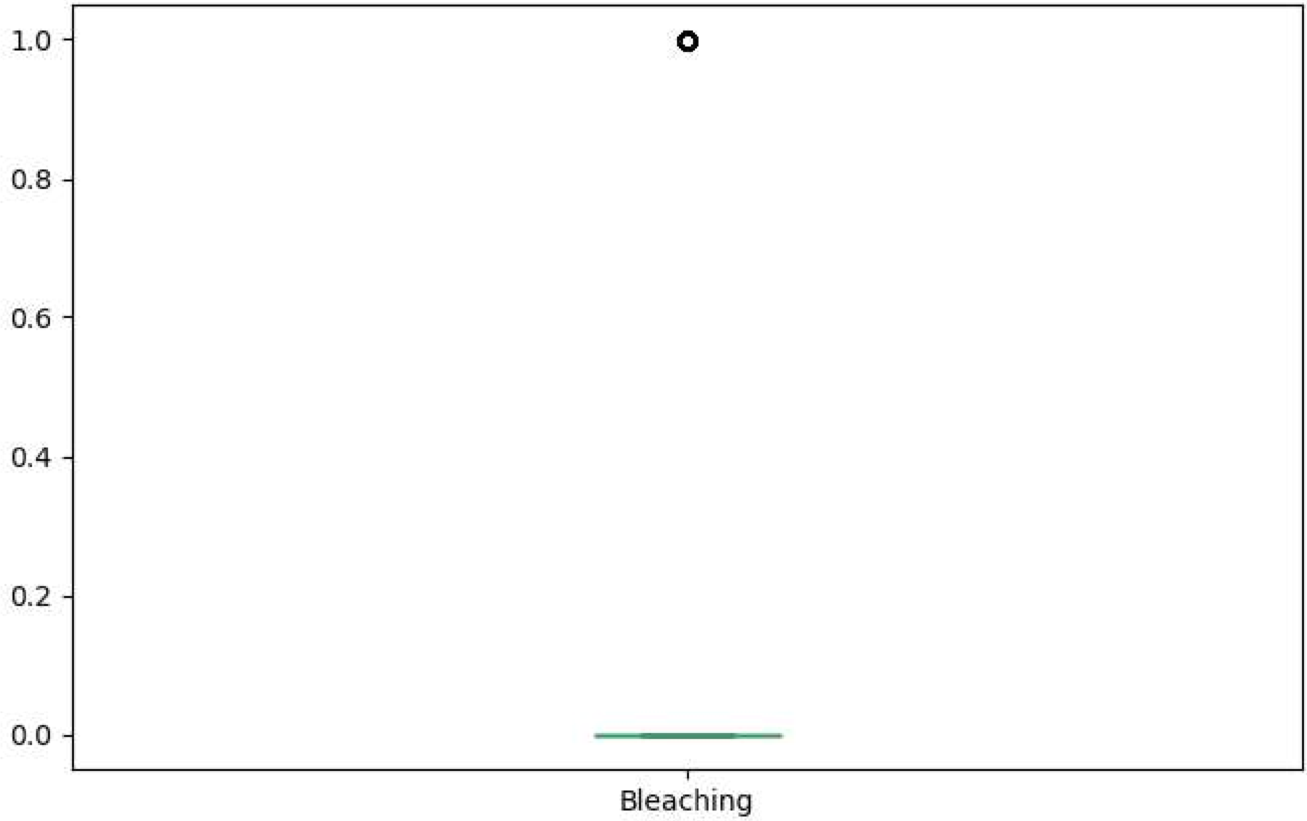

3.0
2.5
2.0
1.5
1.0
0.5
0.0
Sewage

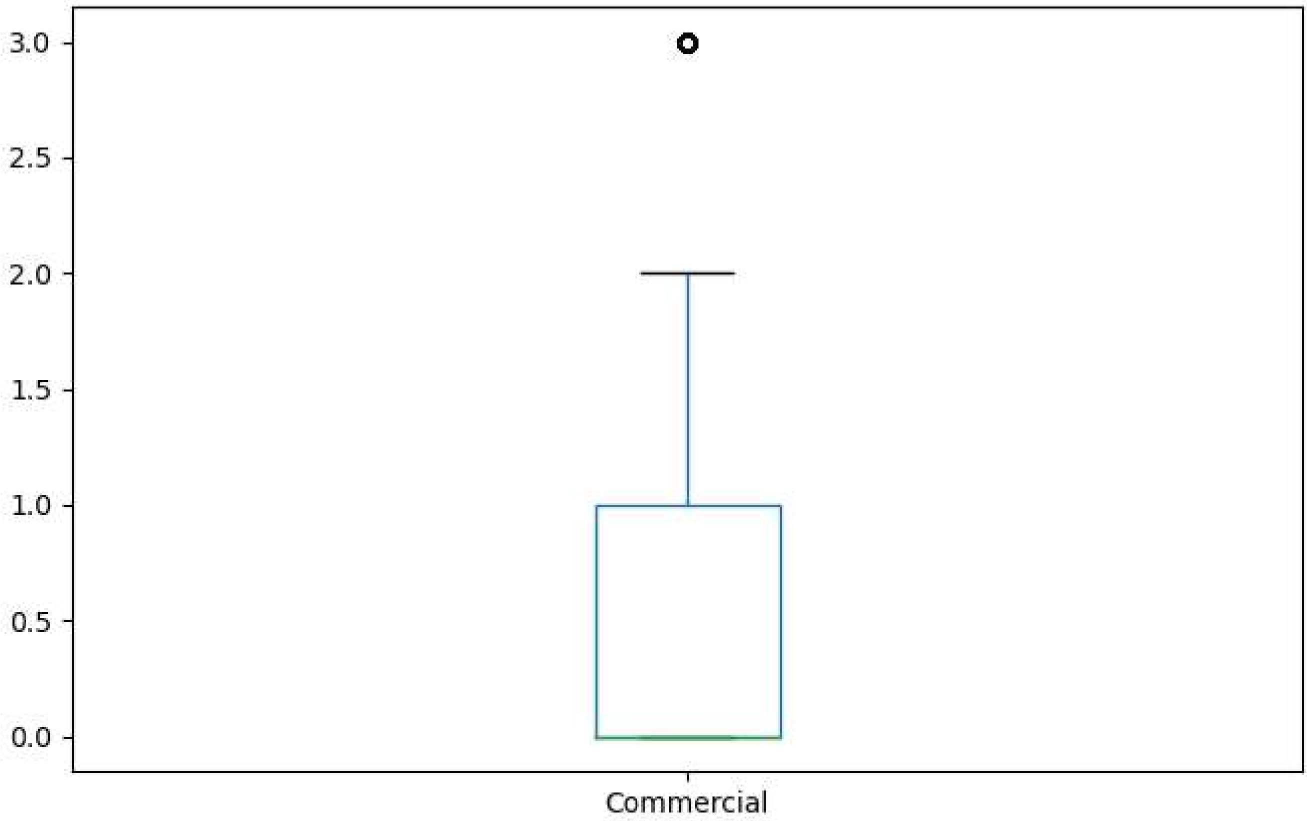

3.0
2.5
2.0
1.5
1.0
0.5
0.0
Siltation



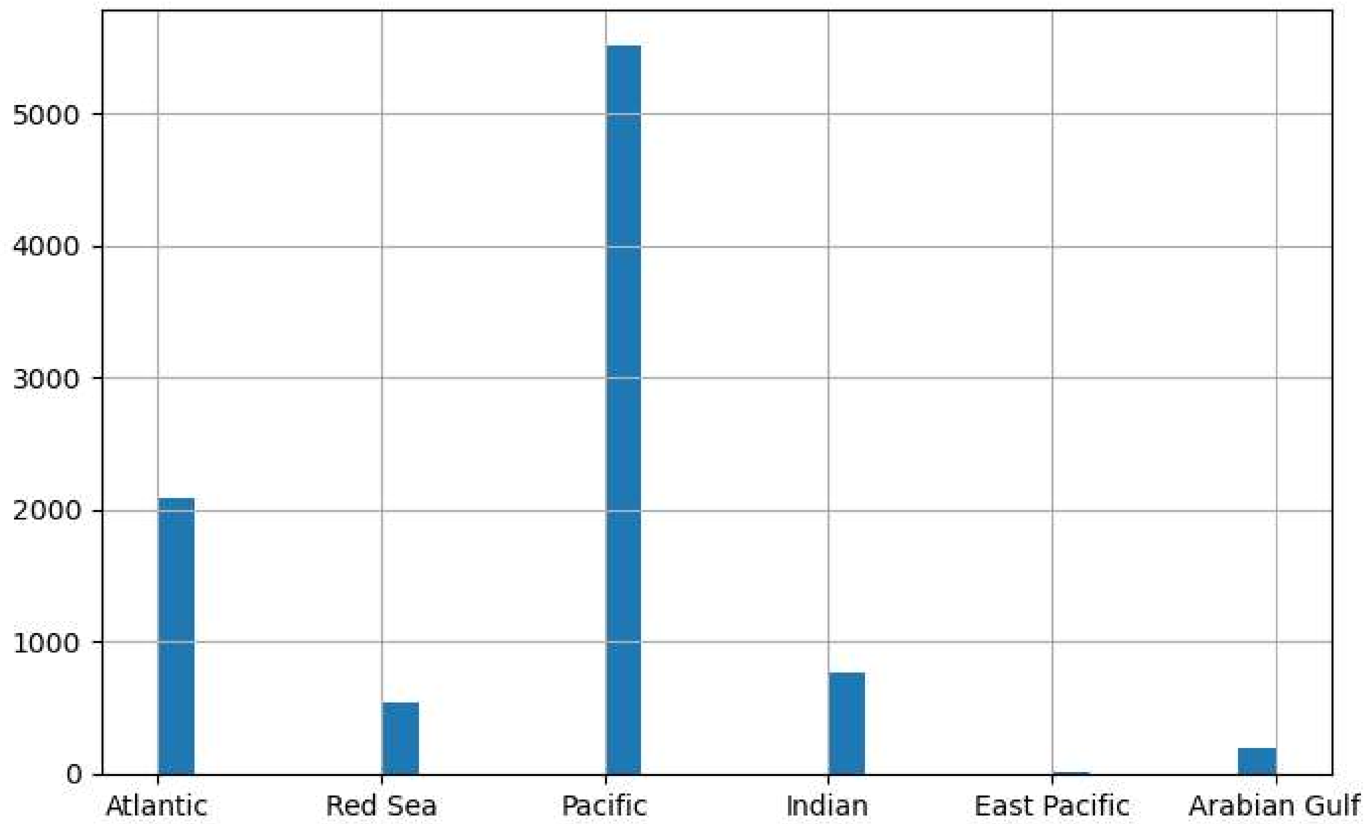
1.0
0.8
0.6
0.4
0.2
0.0
Storms


2017.5
2015.0
2012.5
2010.0
2007.5
2005.0
2002.5
2000.0
1997.5
Year


1.0
0.8
0.6
0.4
0.2
0.0
Bleaching

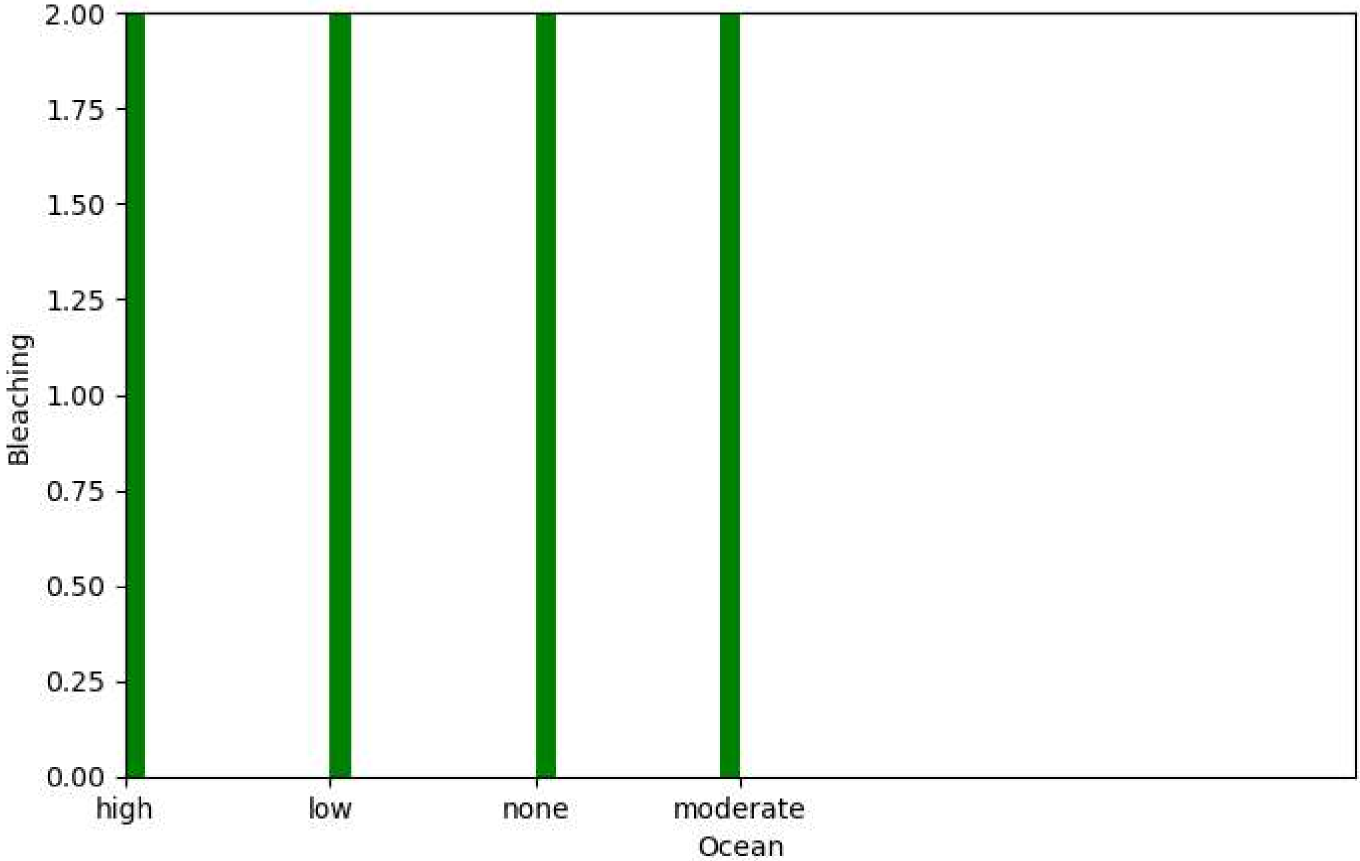

3.0
2.5
2.0
1.5
1.0
0.5
0.0
Commercial

```
#Histogram
df['Ocean'].hist(bins=30, figsize=(8, 5)); # we can specify the number of bins
```



5000
4000
3000
2000
1000
0
Atlantic
Red Sea
Pacific
Indian
East Pacific
Arabian Gulf

```python
ax = df['Human Impact'].hist(bins=30, grid=False, color='green', figsize=(8, 5)) # grid turned off and color changed
ax.set_xlabel('Ocean')
ax.set_ylabel('Bleaching')
ax.set_xlim(0,6) #limiting display range to 0-6 for the x-axis
ax.set_ylim(0,2); #limiting display range to 0-2 for the y-axis
```



2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Bleaching
high
low
none
moderate
Ocean

```python
#Boxplot
df['Depth'].plot.box(figsize=(8, 5)); # Boxplot of a column
```

Data Cleaning

```
#Check if there are missing values in the dataset
df.isnull().sum().sum()
```

```
0
```

```
#Check if there are duplicate rows in the dataset
df.duplicated().sum()
```

```
2412
```

```
#Removing duplicates from the dataset
df.drop_duplicates(keep="first",inplace=True)
```

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

```
0
```

```
#Check if duplicate rows have been removed successfully from the dataset
df.duplicated().sum()
```

```
0
```

Label encoding columns having non-integer values

```
df['Human Impact'].replace({'none':0,'low':1,'moderate':2,'high':3},inplace=True)
```

```
df['Siltation'].replace({'never':0,'occasionally':1,'often':2,'always':3},inplace=True)
```

```
df['Dynamite'].replace({'none':0,'low':1,'moderate':2,'high':3},inplace=True)
```

```
df['Poison'].replace({'none':0,'low':1,'moderate':2,'high':3},inplace=True)
```

```
df['Sewage'].replace({'none':0,'low':1,'moderate':2,'high':3},inplace=True)
```

```
df['Industrial'].replace({'none':0,'low':1,'moderate':2,'high':3},inplace=True)
```

```
df['Commercial'].replace({'none':0,'low':1,'moderate':2,'high':3},inplace=True)
```

```
df['Storms'].replace({'yes':1,'no':0},inplace=True)
```

```
df['Bleaching'].replace({'Yes':1,'No':0},inplace=True)
```

```
df['Ocean'].replace({'Atlantic':0,'Pacific':1,'Red Sea':2,'East Pacific':3,'Arabian Gulf':4,'Indian':5},inplace=True)
```

```
#Data after Label encoding
df.head()
```

	Bleaching	Ocean	Year	Depth	Storms	Human Impact	Siltation	Dynamite	Poison	Sewage	Industrial	Commercial
0	0	0	2005	4.0	1	3	2	0	0	3	0	0
1	0	2	2004	6.0	0	3	1	0	0	1	0	0
2	0	1	1998	3.0	0	1	0	0	0	0	1	0
3	0	1	1998	10.0	0	1	0	0	0	0	1	0
4	0	0	1997	10.0	0	3	0	0	0	3	2	0

```
#No. of rows in the dataset after cleaning
print(len(df.axes[0]))
```

```
6699
```

```
#Data types of dataset columns after label encoding
df.dtypes
```

```
Bleaching         int64
Ocean             int64
Year              int64
Depth           float64
Storms            int64
Human Impact      int64
Siltation         int64
Dynamite          int64
Poison            int64
Sewage            int64
Industrial        int64
Commercial        int64
dtype: object
```

```python
#Statistics for all the dataset columns
df.describe()
```

	Bleaching	Ocean	Year	Depth	Storms	Human Impact	Siltation	Dynamite	Poison	Sewage	Industrial	Comm
count	6699.000000	6699.000000	6699.000000	6699.000000	6699.000000	6699.000000	6699.000000	6699.000000	6699.000000	6699.000000	6699.000000	6699.0
mean	0.032691	1.252127	2007.596955	6.540140	0.537394	1.528586	0.682639	0.206449	0.188237	0.719510	0.256307	0.7
std	0.177841	1.373574	4.857230	3.565411	0.498637	0.816674	0.766896	0.594526	0.526632	0.781233	0.587381	0.9
min	0.000000	0.000000	1997.000000	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.000000	1.000000	2004.000000	3.500000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
50%	0.000000	1.000000	2007.000000	6.000000	1.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.0
75%	0.000000	1.000000	2011.000000	10.000000	1.000000	2.000000	1.000000	0.000000	0.000000	1.000000	0.000000	1.0
max	1.000000	5.000000	2017.000000	23.000000	1.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.0

```python
#Variance
df.var()
```

```
Bleaching         0.031627
Ocean             1.886706
Year             23.592681
Depth            12.712158
Storms            0.248639
Human Impact      0.666957
Siltation         0.588129
Dynamite          0.353461
Poison            0.277341
Sewage            0.610325
Industrial        0.345017
Commercial        0.959156
dtype: float64
```

```
#Skewness
df.skew()
```

```
Bleaching        5.256920
Ocean            1.829616
Year            -0.004985
Depth            0.546363
Storms          -0.150028
Human Impact     0.234592
Siltation        0.980309
Dynamite         3.182245
Poison           3.048365
Sewage           0.969312
Industrial       2.530085
Commercial       0.953451
dtype: float64
```

```
#Kurtosis
df.kurtosis()
```

```
Bleaching        25.642868
Ocean             2.492269
Year             -0.738879
Depth            -0.224054
Storms           -1.978082
Human Impact     -0.551090
Siltation         0.527299
Dynamite          9.937500
Poison            9.282395
Sewage            0.564183
Industrial        6.438009
Commercial       -0.304467
dtype: float64
```

Data Selection

```
#Column-wise correlation in the dataset
df.corr()
```

	Bleaching	Ocean	Year	Depth	Storms	Human Impact	Siltation	Dynamite	Poison	Sewage	Industrial	Commercial
Bleaching	1.000000	0.019426	-0.266985	0.005983	0.022410	0.003330	-0.155989	0.033590	0.021960	-0.001690	0.014105	-0.145389
Ocean	0.019426	1.000000	-0.015267	-0.065196	-0.156436	-0.095398	-0.139602	0.012123	-0.007416	-0.048174	-0.086029	-0.114339
Year	-0.266985	-0.015267	1.000000	-0.012651	0.017566	-0.013053	0.186418	-0.017040	-0.042301	0.041102	0.012456	0.214268
Depth	0.005983	-0.065196	-0.012651	1.000000	-0.019668	-0.051640	-0.054589	0.009205	-0.010656	0.017046	0.037589	0.101664
Storms	0.022410	-0.156436	0.017566	-0.019668	1.000000	0.071524	0.022839	-0.045434	-0.029367	-0.004690	-0.013613	0.037164
Human Impact	0.003330	-0.095398	-0.013053	-0.051640	0.071524	1.000000	0.240948	0.217079	0.219547	0.365566	0.196518	0.188543
Siltation	-0.155989	-0.139602	0.186418	-0.054589	0.022839	0.240948	1.000000	-0.002322	0.022251	0.167877	0.171984	0.239464
Dynamite	0.033590	0.012123	-0.017040	0.009205	-0.045434	0.217079	-0.002322	1.000000	0.716062	0.053333	0.089579	0.177938
Poison	0.021960	-0.007416	-0.042301	-0.010656	-0.029367	0.219547	0.022251	0.716062	1.000000	0.116739	0.111393	0.169661
Sewage	-0.001690	-0.048174	0.041102	0.017046	-0.004690	0.365566	0.167877	0.053333	0.116739	1.000000	0.312208	0.210740
Industrial	0.014105	-0.086029	0.012456	0.037589	-0.013613	0.196518	0.171984	0.089579	0.111393	0.312208	1.000000	0.157977
Commercial	-0.145389	-0.114339	0.214268	0.101664	0.037164	0.188543	0.239464	0.177938	0.169661	0.210740	0.157977	1.000000

```python
#Import seaborn library
import seaborn as sns
```

Pearson correlation

```
sns.heatmap(df.corr('pearson'),annot=True)
```

```
<AxesSubplot:>
```


Figure 1
x= y=
Bleaching 1 0.019 -0.27 0.006 0.022 0.0033 -0.16 0.034 0.022 -0.0017 0.014 -0.15
Ocean 0.019 1 -0.015 -0.065 -0.16 -0.095 -0.14 0.012 -0.0074 -0.048 -0.086 -0.11
Year -0.27 -0.015 1 -0.013 0.018 -0.013 0.19 -0.017 -0.042 0.041 0.012 0.21
Depth 0.006 -0.065 -0.013 1 -0.02 -0.052 -0.055 0.0092 -0.011 0.017 0.038 0.1
Storms 0.022 -0.16 0.018 -0.02 1 0.072 0.023 -0.045 -0.029 -0.0047 -0.014 0.037
Human Impact 0.0033 -0.095 -0.013 -0.052 0.072 1 0.24 0.22 0.22 0.37 0.2 0.19
Siltation -0.16 -0.14 0.19 -0.055 0.023 0.24 1 -0.0023 0.022 0.17 0.17 0.24
Dynamite 0.034 0.012 -0.017 0.0092 -0.045 0.22 -0.0023 1 0.72 0.053 0.09 0.18
Poison 0.022 -0.0074 -0.042 -0.011 -0.029 0.22 0.022 0.72 1 0.12 0.11 0.17
Sewage -0.0017 -0.048 0.041 0.017 -0.0047 0.37 0.17 0.053 0.12 1 0.31 0.21
Industrial 0.014 -0.086 0.012 0.038 -0.014 0.2 0.17 0.09 0.11 0.31 1 0.16
Commercial -0.15 -0.11 0.21 0.1 0.037 0.19 0.24 0.18 0.17 0.21 0.16 1
Bleaching Ocean Year Depth Storms Human Impact Siltation Dynamite Poison Sewage Industrial Commercial
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

```
All the three correlation methods namely Pearson, Spearman and Kendall give varying results.

Target column: Bleaching

Pearson correlation results in:

Columns:
Siltation and Commercial are similar
Poison and Storms are similar

Since,
Siltation is more correlated to Bleaching compared to column Commercial
Storms is more correlated to Bleaching compared to Poison

Thus, columns Commercial and Poison will be dropped to remove redundancy.
```

```python
df.drop(['Commercial','Poison'],axis=1).head()
```

20]:

	Bleaching	Ocean	Year	Depth	Storms	Human Impact	Siltation	Dynamite	Sewage	Industrial
0	0	0	2005	4.0	1	3	2	0	3	0
1	0	2	2004	6.0	0	3	1	0	1	0
2	0	1	1998	3.0	0	1	0	0	0	1
3	0	1	1998	10.0	0	1	0	0	0	1
4	0	0	1997	10.0	0	3	0	0	3	2

```python
df.replace('', numpy.nan, inplace=True)
```

```python
df.dropna(inplace=True)
```

Data Splitting and Model Building (Logistic regression)

Logistic Regression using sklearn

```
#Logistic regression model using sklearn
X = df.iloc[:, 1:]
y = df.iloc[:,0]
```

```
#Split in training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

```
#Scale
from sklearn.preprocessing import StandardScaler
X_sca = StandardScaler()
X_train = X_sca.fit_transform(X_train)
X_test = X_sca.fit_transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
y_pred = clf.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1619,    0],
       [  56,    0]], dtype=int64)
```

```
clf.score(X_test, y_test)
```

```
0.9665671641791045
```

```
#Co-efficients of the Logistic regression equation
clf.coef_
```

```
array([[ 3.24787919e-03, -1.67442219e+00, -2.13546706e-02,
         2.07130561e-01, -5.58679555e-02, -1.35227966e+00,
         2.21634742e-01,  1.12713400e-03,  7.82692715e-02]])
```

```
#y-intercept of the Logistic regression equation
clf.intercept_
```

```
array([-5.49964608])
```

```
Logistic regression equation using Pearson

y -> target variable i.e. Bleaching
a -> y-intercept of Bleaching
b0 -> co-efficient of Ocean
b1 -> co-efficient of Year
b2 -> co-efficient of Depth
b3 -> co-efficient of Storms
b4 -> co-efficient of Human Impact
b5 -> co-efficient of Siltation
b6 -> co-efficient of Dynamite
b7 -> co-efficient of Sewage
b8 -> co-efficient of Industrial

General equation: y = a + b0x0 + b1x1 + ... + bnxn

Actual equation: Bleaching = -5.50 + 3.25(Ocean) - 1.67(Year) - 2.14(Depth) + 2.07(Storms) - 5.59(Human Impact) -
                             1.35(Siltation) + 2.22(Dynamite) + 1.13(Sewage) + 7.83(Industrial)
```

Model evaluation through k-fold cross validation and evaluation metrics

```
#K-fold cross-validation
#Logistic Regression
X = df.iloc[:,1:]
y = df.iloc[:,0]
k = 5
kf = model_selection.KFold(n_splits=k, random_state=None)
model = LogisticRegression(solver= 'liblinear')
result = cross_val_score(model , X, y, cv = kf)
print("Avg accuracy: {}".format(result.mean()))
```

```
Avg accuracy: 0.9673094200394591
```

```
#Root mean square error
import sklearn
sklearn.metrics.mean_squared_error(y_test,y_pred)
```

```
0.03343283582089552
```

```
#R2 score
import sklearn
sklearn.metrics.r2_score(y_test,y_pred)
```

```
-0.03458925262507728
```

Spearman correlation

```python
sns.heatmap(df.corr('spearman'),annot=True)
```

]: <AxesSubplot:>


Figure 1
x= y=
Bleaching 1 0.017 -0.27 0.009 0.022 0.0041 -0.17 0.039 0.033 -0.00053 0.0092 -0.16
Ocean 0.017 1 -0.0014 -0.1 -0.22 -0.076 -0.15 0.092 0.076 -0.078 -0.091 -0.13
Year -0.27 -0.0014 1 -0.0044 0.011 -0.019 0.21 -0.053 -0.072 0.032 0.02 0.24
Depth 0.009 -0.1 -0.0044 1 -0.023 -0.039 -0.049 0.018 0.0021 0.032 0.054 0.12
Storms 0.022 -0.22 0.011 -0.023 1 0.074 0.026 -0.055 -0.051 0.0098 -0.019 0.033
Human Impact 0.0041 -0.076 -0.019 -0.039 0.074 1 0.22 0.17 0.18 0.34 0.18 0.17
Siltation -0.17 -0.15 0.21 -0.049 0.026 0.22 1 -0.0012 -0.014 0.15 0.15 0.26
Dynamite 0.039 0.092 -0.053 0.018 -0.055 0.17 -0.0012 1 0.68 0.065 0.079 0.17
Poison 0.033 0.076 -0.072 0.0021 -0.051 0.18 -0.014 0.68 1 0.12 0.1 0.16
Sewage -0.00053 -0.078 0.032 0.032 0.0098 0.34 0.15 0.065 0.12 1 0.32 0.22
Industrial 0.0092 -0.091 0.02 0.054 -0.019 0.18 0.15 0.079 0.1 0.32 1 0.18
Commercial -0.16 -0.13 0.24 0.12 0.033 0.17 0.26 0.17 0.16 0.22 0.18 1
Bleaching Ocean Year Depth Storms Human Impact Siltation Dynamite Poison Sewage Industrial Commercial
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

```
All the three correlation methods namely Pearson, Spearman and Kendall give varying results.

Target column: Bleaching

Spearman correlation results in:

Columns:
Siltation and Commercial are similar
Depth and Industrial are similar

Since,
Commercial is more correlated to Bleaching compared to column Siltation
Industrial is more correlated to Bleaching compared to Depth

Thus, columns Siltation and Depth will be dropped to remove redundancy.
```

```python
df.drop(['Siltation','Depth'],axis=1).head()
```

93]:

	Bleaching	Ocean	Year	Storms	Human Impact	Dynamite	Poison	Sewage	Industrial	Commercial
0	0	0	2005	1	3	0	0	3	0	0
1	0	2	2004	0	3	0	0	1	0	0
2	0	1	1998	0	1	0	0	0	1	0
3	0	1	1998	0	1	0	0	0	1	0
4	0	0	1997	0	3	0	0	3	2	0

Data Splitting and Model Building (Logistic regression)

Logistic Regression using sklearn

```
#Logistic regression model using sklearn
X = df.iloc[:, 1:]
y = df.iloc[:,0]
```

```
#Split in training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

```
#Scale
from sklearn.preprocessing import StandardScaler
X_sca = StandardScaler()
X_train = X_sca.fit_transform(X_train)
X_test = X_sca.fit_transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
y_pred = clf.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1618,    0],
       [  57,    0]], dtype=int64)
```

```python
clf.score(X_test, y_test)
```

```
0.9659701492537314
```

```python
#Co-efficients of the Logistic regression equation
clf.coef_
```

```
array([[-0.08315876, -1.76831506,  0.10950687, -0.03091513,  0.18976625,
        -0.02486071,  0.15358178,  0.05796166, -1.89370559]])
```

```python
#y-intercept of the Logistic regression equation
clf.intercept_
```

```
array([-5.92543204])
```

```
Logistic regression equation using Pearson

y -> target variable i.e. Bleaching
a -> y-intercept of Bleaching
b0 -> co-efficient of Ocean
b1 -> co-efficient of Year
b2 -> co-efficient of Storms
b3 -> co-efficient of Human Impact
b4 -> co-efficient of Dynamite
b5 -> co-efficient of Poison
b6 -> co-efficient of Sewage
b7 -> co-efficient of Industrial
b8 -> co-efficient of Commercial

General equation: y = a + b0x0 + b1x1 + ... + bnxn

Actual equation: Bleaching = -5.93 - 0.08(Ocean) - 1.77(Year) + 0.11(Storms) - 0.03(Human Impact) + 0.20(Dynamite) -
                             0.02(Poison) + 0.15(Sewage) + 0.06(Industrial) - 1.90(Commercial)
```

Model evaluation through k-fold cross validation and evaluation metrics

```
#K-fold cross-validation
#Logistic Regression
X = df.iloc[:,1:]
y = df.iloc[:,0]
k = 5
kf = model_selection.KFold(n_splits=k, random_state=None)
model = LogisticRegression(solver= 'liblinear')
result = cross_val_score(model , X, y, cv = kf)
print("Avg accuracy: {}".format(result.mean()))
```

```
Avg accuracy: 0.966712405114086
```

```
#Root mean square error
import sklearn
sklearn.metrics.mean_squared_error(y_test,y_pred)
```

```
0.03402985074626866
```

```
#R2 score
import sklearn
sklearn.metrics.r2_score(y_test,y_pred)
```

```
-0.03522867737948099
```

Kendall correlation

```
sns.heatmap(df.corr('kendall'),annot=True)
```

]: <AxesSubplot:>

Figure 1
x= y=
Bleaching 1 0.016 -0.22 0.0076 0.022 0.0038 -0.17 0.039 0.032 -0.0005 0.009 -0.15
Ocean 0.016 1 -0.00065 -0.082 -0.21 -0.067 -0.14 0.085 0.071 -0.07 -0.084 -0.11
Year -0.22 -0.00065 1 -0.0029 0.0096 -0.015 0.17 -0.044 -0.059 0.026 0.016 0.19
Depth 0.0076 -0.082 -0.0029 1 -0.019 -0.031 -0.039 0.015 0.0019 0.026 0.045 0.092
Storms 0.022 -0.21 0.0096 -0.019 1 0.069 0.025 -0.054 -0.05 0.0093 -0.019 0.031
Human Impact 0.0038 -0.067 -0.015 -0.031 0.069 1 0.2 0.16 0.17 0.31 0.16 0.15
Siltation -0.17 -0.14 0.17 -0.039 0.025 0.2 1 -0.001 -0.014 0.14 0.14 0.24
Dynamite 0.039 0.085 -0.044 0.015 -0.054 0.16 -0.001 1 0.67 0.06 0.076 0.16
Poison 0.032 0.071 -0.059 0.0019 -0.05 0.17 -0.014 0.67 1 0.11 0.099 0.15
Sewage -0.0005 -0.07 0.026 0.026 0.0093 0.31 0.14 0.06 0.11 1 0.3 0.2
Industrial 0.009 -0.084 0.016 0.045 -0.019 0.16 0.14 0.076 0.099 0.3 1 0.17
Commercial -0.15 -0.11 0.19 0.092 0.031 0.15 0.24 0.16 0.15 0.2 0.17 1
Bleaching Ocean Year Depth Storms Human Impact Siltation Dynamite Poison Sewage Industrial Commercial
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

```
All the three correlation methods namely Pearson, Spearman and Kendall give varying results.

Target column: Bleaching

Kendall correlation results in:

Columns:
Siltation and Commercial are similar
Dynamite and Poison are similar

Since,
Commercial is more correlated to Bleaching compared to column Siltation
Dynamite is more correlated to Bleaching compared to Poison

Thus, columns Siltation and Poison will be dropped to remove redundancy.
```

```python
df.drop(['Siltation','Poison'],axis=1).head()
```

0]:

	Bleaching	Ocean	Year	Depth	Storms	Human Impact	Dynamite	Sewage	Industrial	Commercial
0	0	0	2005	4.0	1	3	0	3	0	0
1	0	2	2004	6.0	0	3	0	1	0	0
2	0	1	1998	3.0	0	1	0	0	1	0
3	0	1	1998	10.0	0	1	0	0	1	0
4	0	0	1997	10.0	0	3	0	3	2	0

Data Splitting and Model Building (Logistic regression)

Logistic Regression using sklearn

```
#Logistic regression model using sklearn
X = df.iloc[:, 1:]
y = df.iloc[:,0]
```

```
#Split in training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

```
#Scale
from sklearn.preprocessing import StandardScaler
X_sca = StandardScaler()
X_train = X_sca.fit_transform(X_train)
X_test = X_sca.fit_transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
y_pred = clf.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1626,    1],
       [  48,    0]], dtype=int64)
```

```
clf.score(X_test, y_test)
```

```
0.9707462686567164
```

```
#Co-efficients of the Logistic regression equation
clf.coef_
```

```
array([[-0.1330662 , -1.70696756,  0.04019537,  0.27499422, -0.1259618 ,
         0.26575067,  0.16577799,  0.04674627, -2.45702874]])
```

```
#y-intercept of the Logistic regression equation
clf.intercept_
```

```
array([-6.22955999])
```

```
Logistic regression equation using Pearson

y -> target variable i.e. Bleaching
a -> y-intercept of Bleaching
b0 -> co-efficient of Ocean
b1 -> co-efficient of Year
b2 -> co-efficient of Depth
b3 -> co-efficient of Storms
b4 -> co-efficient of Human Impact
b5 -> co-efficient of Dynamite
b6 -> co-efficient of Sewage
b7 -> co-efficient of Industrial
b8 -> co-efficient of Commercial

General equation: y = a + b0x0 + b1x1 + ... + bnxn

Actual equation: Bleaching = -6.23 - 0.13(Ocean) - 1.71(Year) + 0.04(Depth) + 0.27(Storms) - 0.13(Human Impact) +
                             0.27(Dynamite) + 0.17(Sewage) + 0.05(Industrial) - 2.46(Commercial)
```

Model evaluation through k-fold cross validation and evaluation metrics

```
#K-fold cross-validation
#Logistic Regression
X = df.iloc[:,1:]
y = df.iloc[:,0]
k = 5
kf = model_selection.KFold(n_splits=k, random_state=None)
model = LogisticRegression(solver= 'liblinear')
result = cross_val_score(model , X, y, cv = kf)
print("Avg accuracy: {}".format(result.mean()))
```

```
Avg accuracy: 0.9670109125767725
```

```
#Root mean square error
import sklearn
sklearn.metrics.mean_squared_error(y_test,y_pred)
```

```
0.029253731343283584
```

```
#R2 score
import sklearn
sklearn.metrics.r2_score(y_test,y_pred)
```

```
-0.0509501126818277
```

```
Comparing the accuracy given by the three methods i.e. Pearson, Spearman and Kendall through Logistic regression using sklearn:

Pearson: 0.9665671641791045
Spearman: 0.9659701492537314
Kendall: 0.9707462686567164

Thus, Kendall correlation along with Logistic Regression model gives the most accurate results of the three on evaluation.
```

```
Comparing the average accuracy given by the three methods i.e. Pearson, Spearman and Kendall through K-fold cross validation:

Pearson: 0.9673094200394591
Spearman: 0.966712405114086
Kendall: 0.9670109125767725

Thus, Pearson correlation along with Logistic Regression model gives the most accurate results of the three on evaluation.
```

```
Comparing the RMSE (Root Mean Squar Error) for the three methods i.e. Pearson, Spearman and Kendall:

Pearson: 0.03343283582089552
Spearman: 0.03402985074626866
Kendall: 0.029253731343283584

Thus, Kendall correlation gives the least RMSE evaluation metric.
```

```
Comparing the R2 score for the three methods i.e. Pearson, Spearman and Kendall:

Pearson: -0.03458925262507728
Spearman: -0.03522867737948099
Kendall: -0.0509501126818277

Thus, Pearson correlation gives the best R2 score evaluation metric.
```