jupyter Data exploration ML assignment Last Checkpoint: 14 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [1]:
```
%matplotlib
```

Using matplotlib backend: Qt5Agg

In [2]:
```
from numpy import arange
from matplotlib import pyplot as plt
from scipy.stats import norm
import pandas as pd
```

In [3]:
```
plt.rcParams['figure.figsize'] = [16, 7]
```

In [4]:
```
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")
```

In [5]:
```
df.columns = columns
df.head()
```

Out[5]:

	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

```
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
```

```
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
```

```
df.memory_usage() # in bytes
```

```
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
```

```
df.memory_usage().sum()
```

```
874784
```

```
human_impact_type=('high','low','moderate','none')
human_impact=pd.DataFrame(human_impact_type,columns=['Human Impact'])
human_impact['Human Impact']=human_impact['Human Impact'].astype('category')
human_impact['Human impact']=human_impact['Human Impact'].cat.codes
human_impact
```

	Human Impact	Human impact
0	high	0
1	low	1
2	moderate	2
3	none	3

```
siltation_type=("never","occasionally","often","always")
siltation=pd.DataFrame(siltation_type,columns=['Siltation'])
siltation['Siltation']=siltation['Siltation'].astype('category')
siltation['siltation']=siltation['Siltation'].cat.codes
siltation
```

	Siltation	siltation
0	never	1
1	occasionally	2
2	often	3
3	always	0

```
dynamite_type=('high','low','moderate','none')
dynamite=pd.DataFrame(human_impact_type,columns=['Dynamite'])
dynamite['Dynamite']=dynamite['Dynamite'].astype('category')
dynamite['dynamite']=dynamite['Dynamite'].cat.codes
dynamite
```

	Dynamite	dynamite
0	high	0
1	low	1
2	moderate	2
3	none	3

```
# To get statistics for all the columns at the same time
df.describe()
```

	Year	Depth
count	9111.000000	9111.000000
mean	2007.424212	6.455845
std	4.870591	3.530270
min	1997.000000	0.500000
25%	2004.000000	3.000000
50%	2007.000000	6.000000
75%	2011.000000	10.000000
max	2017.000000	23.000000

```
#### Statistical moments
"""
1. Mean (1st moment)
2. Variance (2nd moment)
3. Skewness (3rd moment)
4. Kurtosis (4th moment)"""
```

```
# Applying mean() to the dataframe returns mean of each column (pandas series)
df.mean()
```

```
Year      2007.424212
Depth        6.455845
dtype: float64
```

```
df['Depth'].mean() # returns the mean of 'Depth' column
```

```
6.455844583470532
```

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

```
Year      23.722658
Depth     12.462809
dtype: float64
```

```python
#### Skewness
"""Skewness is the measure of the symmetry of a distribution compared to standard normal distribution
+ive - right skewed (mean is to the right of mode/median). Long tail in the +ive direction.
0 - symmetric
-ive - left skewed (mean is to the left of mode/median). Long tail in the -ive direction.
```

```python
# skewness
df.skew()
```

```
Year     -0.003448
Depth     0.506123
dtype: float64
```

```python
#### Kurtosis
"""Kurtosis is a measure of the flatness or peakedness of a distribution compared to the normal distribution.
+ive - Leptokurtosis (sharper/spikier peak compared to the normal dist.)
0 - Mesokurtic (normal dist.)
-ive - Platykurtic (flatter peak compared to the normal dist.) eg. Uniform distribution
```

```python
# skewness
df.kurtosis()
```

```
Year     -0.711181
Depth    -0.349015
dtype: float64
```

```python
#### min / max / median
```

```python
# min of each column
df.min()
```

```
Bleaching                  No
Ocean            Arabian Gulf
Year                     1997
Depth                     0.5
Storms                     no
Human Impact             high
Siltation              always
Dynamite                 high
Poison                   high
Sewage                   high
Industrial               high
Commercial               high
dtype: object
```

```python
# max of each column
df.max()
```

```
Bleaching             Yes
Ocean             Red Sea
Year                 2017
Depth                  23
Storms                yes
Human Impact         none
Siltation           often
Dynamite             none
Poison               none
Sewage               none
Industrial           none
Commercial           none
dtype: object
```

```python
# median of each column
df.median()
```

```
Year      2007.0
Depth        6.0
dtype: float64
```

```
#### Correlation
df.corr()
```

	Year	Depth
Year	1.00000	-0.03332
Depth	-0.03332	1.00000

```
import seaborn as sns
```

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

```
<AxesSubplot:>
```

```
### Lineplot
# Plotting with index along the x-axis
df['Year'].plot(figsize=(12, 5), color='black') # color and figsize changed
plt.xlim(1970,2017,1) # range for x-axis
plt.ylim(0,1) # range for x-axis
plt.xlabel('Year')
plt.ylabel('Bleaching'); # \";\" prevents object info from displaying
```

```
### Scatterplot
# plotting one variable against the other
df.plot.scatter('Year', 'Bleaching', figsize=(8, 5))
# The x and y labels are automatically taken from the column names
```

```
<AxesSubplot:xlabel='Year', ylabel='Bleaching'>
```

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

```
# Box plot of all the columns with numerical data
df.boxplot(figsize=(16, 5)) # or df.plot.box()
```

```
<AxesSubplot:xlabel='index', ylabel='Bleaching'>
```

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

```
ax = df['Human Impact'].hist(bins=30, grid=False, color='green', figsize=(8, 5)) # grid turned off and color changed
ax.set_xlabel('Year')
ax.set_ylabel('Bleaching')
ax.set_xlim(0, 70) # limiting display range to 0-70 for the x-axis
ax.set_ylim(0, 120); # limiting display range to 0-120 for the y-axis
```

```
### Barplot
#The bar charts are used to visualize categorical data (nominal or ordinal values) and the height shows the value it represents
```

```
df_avg_BP = df.groupby('Year')['Depth'].mean()
df_avg_BP[:10].plot.bar(color='orange');
```

```
### Multiple Plots
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
# or fig, (ax1, ax2, ax3, ax4) = plt.subplots(2, 2, figsize=(12, 8))
# axes is the axes object(s). It can be a single object or an array of objects.
# In this case, it is an array of dimension 2-by-2
df['Year'].plot(ax = axes[0][0], style='.', color='red') # top left
df['Depth'].plot(ax = axes[0][1], style='.', color='blue') # top right
#df['Industrial'].plot.hist(bins=30, ax = axes[1][0], color='black') # bottom left
#df['Sewage'].plot.hist(bins=20, ax = axes[1][1], color='gray') # bottom right
axes[0][0].set_xlabel('index')
axes[0][1].set_xlabel('index')
#axes[1][0].set_xlabel('Industrial')
#axes[1][1].set_xlabel('Sewage')
axes[0][0].set_ylabel('Year')
axes[0][1].set_ylabel('Depth')
axes[0][0].set_ylim(20, 120)
axes[0][1].set_ylim(20, 240)
#axes[1][0].set_xlim(0, 60)
#axes[1][1].set_xlim(20, 80)
fig.tight_layout()
```

```
### Data cleansing
#Unclean data renders only useless and inaccurate models. Garbage-in, garbage-out (GIGO)
#It is meaningless to spend any time in modeling, if your data is not clean.
#Data cleaning is the most important task in the entire predicive modeling work flow.
#Clean data is critical for training models to achieve good predictive power.
#Data scientists usually spend 70% of their time in understanding and cleaning the data which shows the seriousness of the task c
#On contrary to the common misconception that modeling is the most time consuming task, it just involves 30% of the work.
#Data integrity is questionable when any of the following exists in the data
#Missing values (NaNs),
#Infinite values,
#Outliers,
#Erroneous values and
#Values in different format.
#For each type, we need to apply suitable method(s) to clean the data.
#Let's discuss different methods and techniques to clean the data.
```

Figure 1
1
-0.033
0.021
-0.033
1
-0.05
0.021
-0.05
1
Year
Depth
Human Impact
Year
Depth
Human Impact
1.0
0.8
0.6
0.4
0.2
0.0

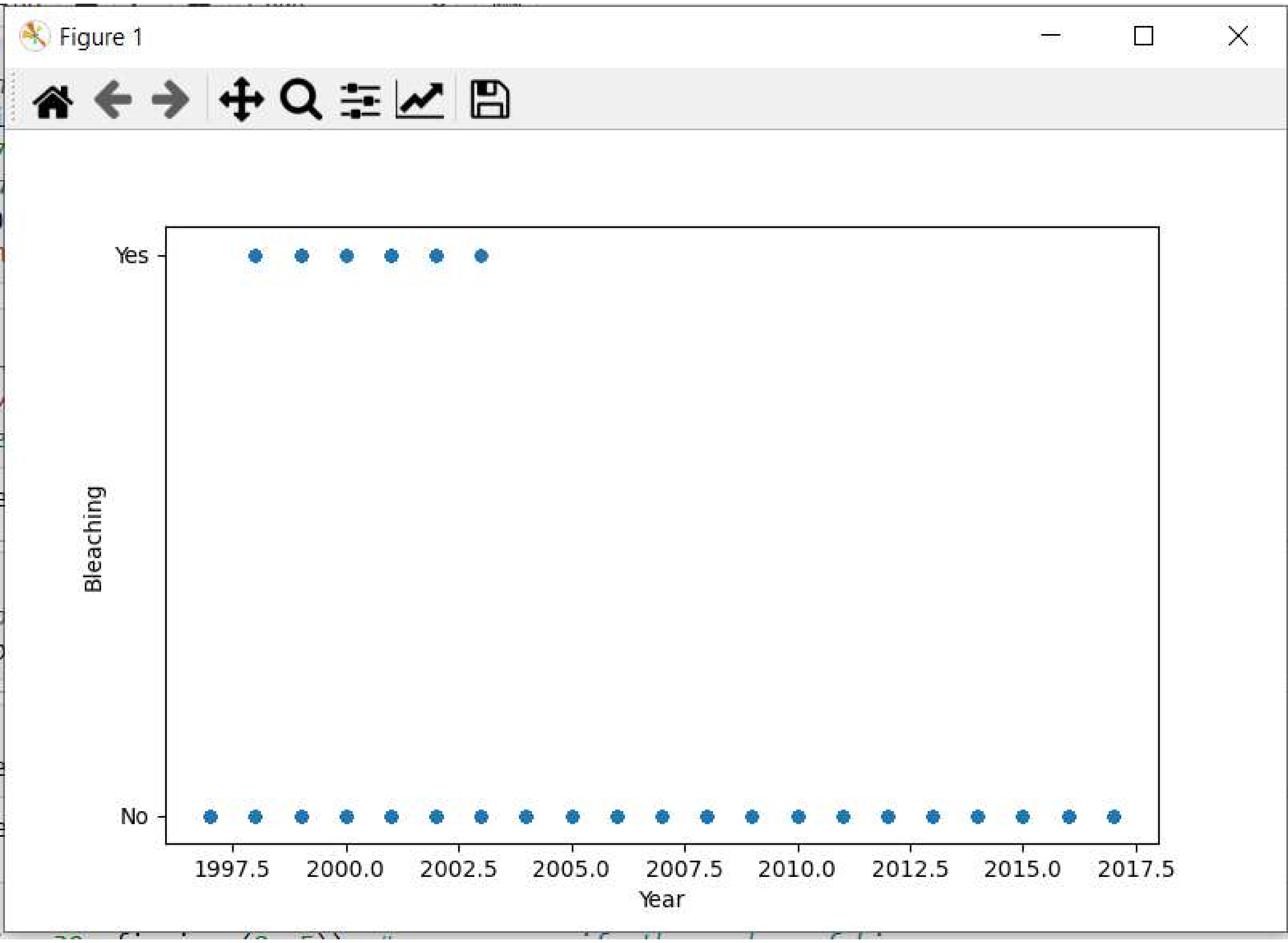
Figure 1
Yes
No
Bleaching
1997.5
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
Year

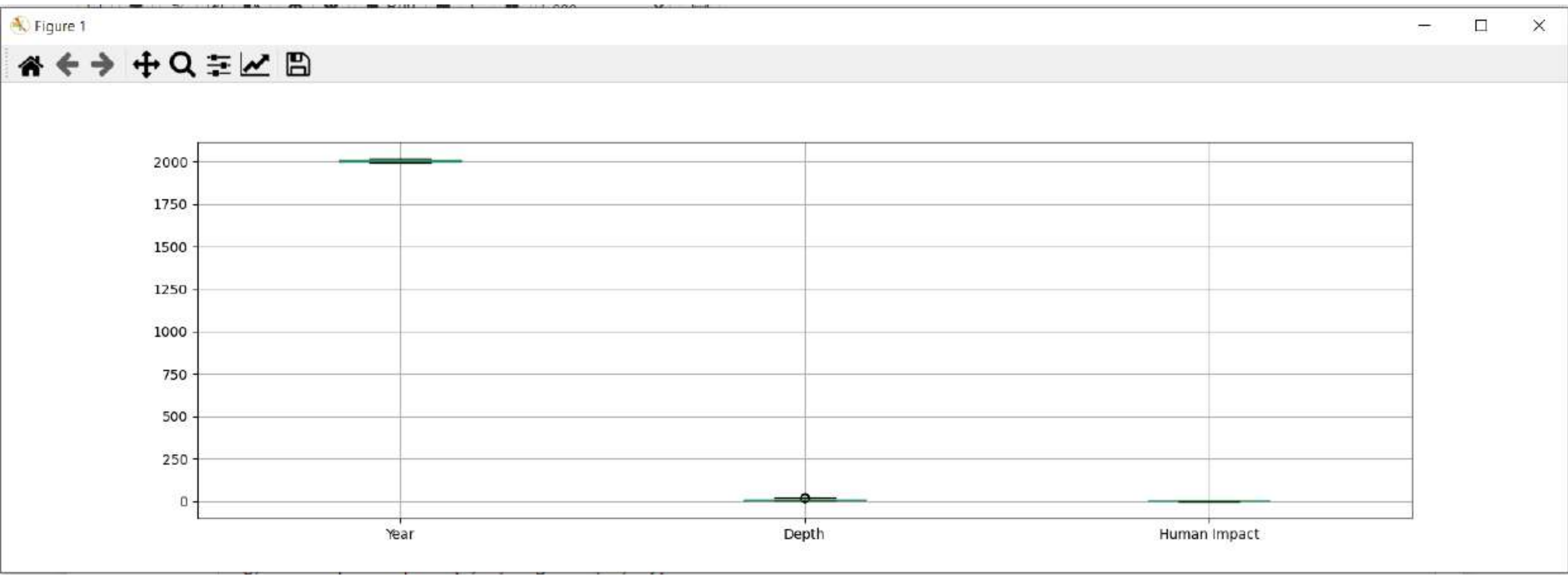
Figure 1
2000
1750
1500
1250
1000
750
500
250
0
Year
Depth
Human Impact