practiceQuestions3(Answers)

October 19, 2023

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
#Reading data from the csv file
water = pd.read_csv("water_portability.csv")
```

[3]:
```
#Displaying the first five rows
water.head()
```

[3]:

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

	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	10.379783	86.990970	2.963135	0
1	15.180013	56.329076	4.500656	0
2	16.868637	66.420093	3.055934	0
3	18.436524	100.341674	4.628771	0
4	11.558279	31.997993	4.075075	0

[4]:
```
#1.
water.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
```

[5]:
```
water.describe()
```

[5]:

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

	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	426.205111	14.284970	66.396293	3.966786	0.390110
std	80.824064	3.308162	16.175008	0.780382	0.487849
min	181.483754	2.200000	0.738000	1.450000	0.000000
25%	365.734414	12.065801	55.844536	3.439711	0.000000
50%	421.884968	14.218338	66.622485	3.955028	0.000000
75%	481.792304	16.557652	77.337473	4.500320	1.000000
max	753.342620	28.300000	124.000000	6.739000	1.000000

[6]:
```
#Checking for missing values
missing = water.isnull().sum()
print("Missing values are:\n", missing)
```

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

[8]:
```
#Checking for duplicate rows on all columns and displaying them
waterDuplicate = water.duplicated(keep=False)
print(waterDuplicate)
```

```
0       False
1       False
2       False
3       False
4       False
        …
3271    False
3272    False
3273    False
3274    False
3275    False
Length: 3276, dtype: bool
```

Thus, there are no duplicates.

[15]:
```
#Filling missing values with the mean of the columns
water['ph'].fillna(water['ph'].mean(), inplace=True)
```

[16]:
```
water['Sulfate'].fillna(water['Sulfate'].mean(), inplace=True)
```

[17]:
```
water['Trihalomethanes'].fillna(water['Trihalomethanes'].mean(), inplace=True)
```

[18]:
```
#Checking for missing values
missing = water.isnull().sum()
print("Missing values are:\n", missing)
```

```
Missing values are:
 ph                 0
Hardness            0
Solids              0
Chloramines         0
Sulfate             0
Conductivity        0
Organic_carbon      0
Trihalomethanes     0
Turbidity           0
Potability          0
dtype: int64
```

[19]:
```
#Identifying categorical features
water.dtypes
```

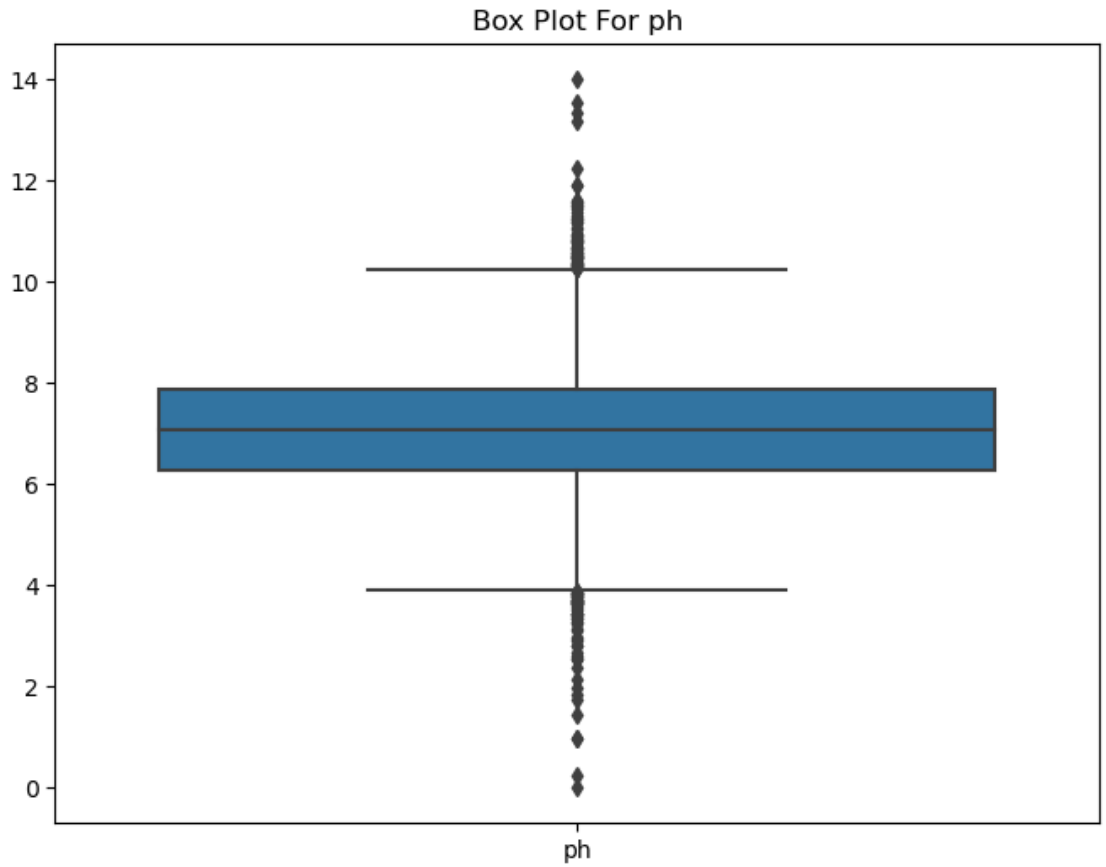
[19]:
```
ph                 float64
Hardness           float64
Solids             float64
Chloramines        float64
Sulfate            float64
Conductivity       float64
Organic_carbon     float64
```

```
Trihalomethanes    float64
Turbidity          float64
Potability           int64
dtype: object
```

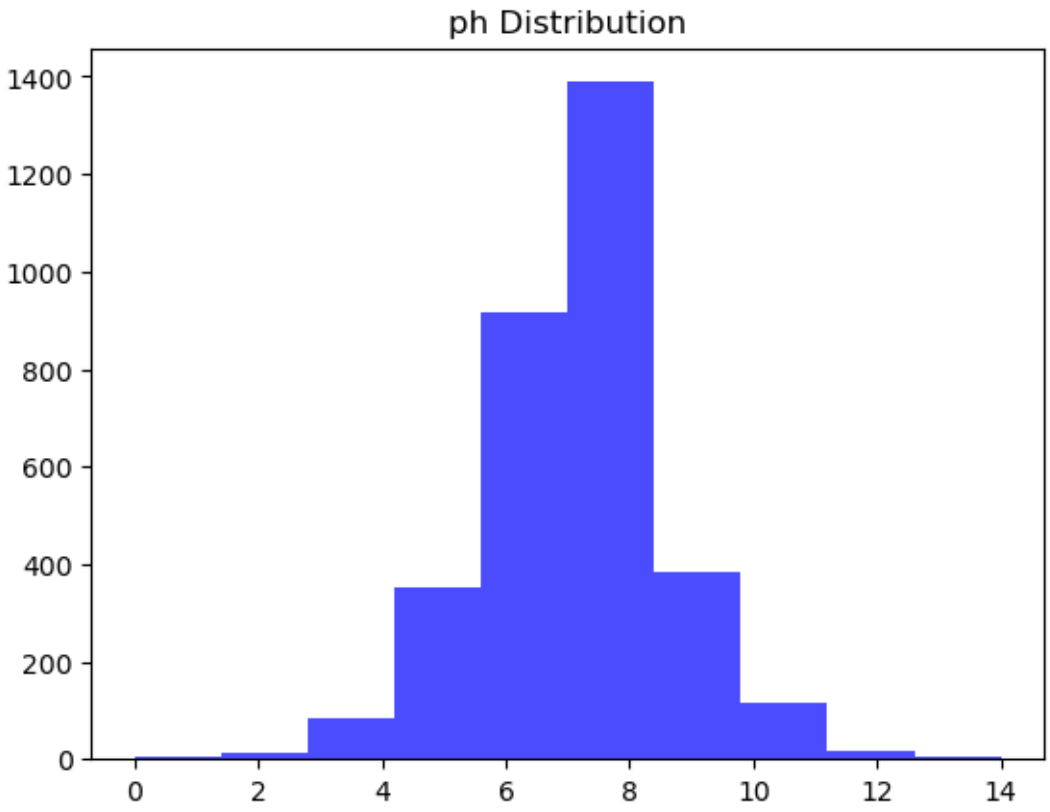
[41]:
```
#Checking for outliers
Q1 = water.quantile(0.25)
Q3 = water.quantile(0.75)
interquartileRange = Q3 - Q1

lowerBound = Q1 - 1.5 * interquartileRange
upperBound = Q3 + 1.5 * interquartileRange
outliers = (water < lowerBound) | (water > upperBound)

#Print columns with outliers
columnOutliers = outliers.any()
print("Columns with outliers:\n")
print(columnOutliers[columnOutliers].index)
```

```
Columns with outliers:

Index(['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
       'Organic_carbon', 'Trihalomethanes', 'Turbidity'],
      dtype='object')
```

[34]:
```
import matplotlib.pyplot as plt
import seaborn as sns
#Using boxplot to visualize the data
plt.figure(figsize=(8, 6))
sns.boxplot(data=water[['ph']])
plt.title("Box Plot For ph")
plt.show()
```



[43]:
```
#Utilizing histogram to visualize the data
plt.hist(water['ph'], bins=10, color='blue', alpha=0.7)
plt.title("ph Distribution")
plt.show()
```



2. An outlier is a datapoint that is far from the cluster of other datapoints in a dataset. The importance of outliers are:

i. It usually indicate errors in data collection which allows for easier identification and correction of these errors.
ii. Outliers also indicate a distortion in statistical analysis, which may lead to users making wrong conclusions. This also allows users to correct these distortions by removing them, ensuring that statistical measures are accurate
iii. They affect the accuracy of machine learning models. This creates a need for outliers to be detected and removed as they can lead to biased reults.
iv. By identifying, analyzing and editing outliers, analysts can make accurate decisions based on the edited data.
v. Anomalities may be identified using outliers, which may be of interest to the analyst.

3.

i. Data Visualization: To find values that are distant from the rest of the data, this method involves visualizing the data using box plots, scatter plots, or histograms. While this approach can be efficient in locating anomalies, it might not be as accurate as other approaches.
ii. Statistical Tests: Statistical tests, like z-scores, are used in this method to find values that stray considerably from the dataset.
iii. Sorting: This entails arranging the data in either ascending or descending order and identifying values that are too high or too low compared to the rest of the data.