dsbda-practice

May 31, 2023

[119]:
```
# Just for using dataset present in drive folder
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

1 Assignment 1 (Data Wrangling)

1. Import all the required Python libraries:

[120]:
```
#import all required libraries

import pandas as pd
```

2. Load the dataset into pandas' data frame:

[121]:
```
df = pd.read_csv("/content/drive/MyDrive/dsbda_datasets/StudentsPerformance.
↪csv")
```

[122]:
```
df
```

[122]:
```
     gender race/ethnicity parental level of education         lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male        group A          associate's degree  free/reduced
4      male        group C                some college      standard
..      …              …                           …           …
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none        72.0           72.0             74
1                 completed        69.0           90.0             88
2                      none        90.0           95.0             93
```

```
3                 none       47.0          57.0            44
4                 none       76.0          78.0            75
..                 ...        ...           ...           ...
995          completed       88.0          99.0            95
996               none       62.0          55.0            55
997          completed       59.0          71.0            65
998          completed       68.0          78.0            77
999               none       77.0          86.0            86

[1000 rows x 8 columns]
```

3. Data Preprocessing:

[123]:
```
# Checking for missing values

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

[123]:
```
gender                         0
race/ethnicity                 0
parental level of education    3
lunch                          0
test preparation course        0
math score                     1
reading score                  2
writing score                  0
dtype: int64
```

[124]:
```
# initial statistics

df.describe()
```

[124]:
```
       math score  reading score  writing score
count  999.000000     998.000000    1000.000000
mean    66.093093      69.178357      68.054000
std     15.170122      14.611940      15.195657
min      0.000000      17.000000      10.000000
25%     57.000000      59.000000      57.750000
50%     66.000000      70.000000      69.000000
75%     77.000000      79.000000      79.000000
max    100.000000     100.000000     100.000000
```

[125]:
```
# first few rows of dataset

df.head()
```

[125]:
```
   gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
```

```
1  female         group C              some college      standard
2  female         group B           master's degree      standard
3    male         group A        associate's degree  free/reduced
4    male         group C              some college      standard

  test preparation course  math score  reading score  writing score
0                    none        72.0           72.0             74
1               completed        69.0           90.0             88
2                    none        90.0           95.0             93
3                    none        47.0           57.0             44
4                    none        76.0           78.0             75
```

[126]:
```
# last few rows of dataset

df.tail()
```

[126]:
```
     gender race/ethnicity parental level of education         lunch  \
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
995               completed        88.0           99.0             95
996                    none        62.0           55.0             55
997               completed        59.0           71.0             65
998               completed        68.0           78.0             77
999                    none        77.0           86.0             86
```

[127]:
```
# Get information about the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  997 non-null    object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   999 non-null    float64
 6   reading score                998 non-null    float64
 7   writing score                1000 non-null   int64
```

```
dtypes: float64(2), int64(1), object(5)
memory usage: 62.6+ KB
```

[128]:
```
# Dimensions of dataframe

df.shape
```

[128]:
```
(1000, 8)
```

[129]:
```
# Variable Descriptions

variable_descriptions = {
    'gender': 'Gender of the person',
    'race/ethnicity': 'Caste of the person',
    # Add more variable descriptions as needed
}

for column in df.columns:
  print(column +" : "+ variable_descriptions.get(column, 'No description␣
 ↪available'))
```

```
gender : Gender of the person
race/ethnicity : Caste of the person
parental level of education : No description available
lunch : No description available
test preparation course : No description available
math score : No description available
reading score : No description available
writing score : No description available
```

4. Data Formatting:

[130]:
```
df.dtypes
```

[130]:
```
gender                          object
race/ethnicity                  object
parental level of education     object
lunch                           object
test preparation course         object
math score                     float64
reading score                  float64
writing score                    int64
dtype: object
```

[131]:
```
df['math score'] = df['math score'].astype(str)
```

[132]:
```
df.dtypes
```

[132]:
```
gender                         object
race/ethnicity                 object
parental level of education    object
lunch                          object
test preparation course        object
math score                     object
reading score                 float64
writing score                   int64
dtype: object
```

4. Data Normalization

[133]:
```
from sklearn.preprocessing import MinMaxScaler
```

[134]:
```
columns_to_normalize = ['math score']
```

[135]:
```
scaler = MinMaxScaler()
```

[136]:
```
df[columns_to_normalize] = scaler.fit_transform(df[columns_to_normalize])
```

[137]:
```
df.head()
```

[137]:
```
   gender race/ethnicity parental level of education         lunch  \
0  female        group B           bachelor's degree      standard
1  female        group C                some college      standard
2  female        group B             master's degree      standard
3    male        group A          associate's degree  free/reduced
4    male        group C                some college      standard

  test preparation course  math score  reading score  writing score
0                    none        0.72           72.0             74
1               completed        0.69           90.0             88
2                    none        0.90           95.0             93
3                    none        0.47           57.0             44
4                    none        0.76           78.0             75
```

2 Assignment 2 (Data Wrangling)

1. Import all the required Python Libraries.

[138]:
```
import pandas as pd
```

2. Load the Dataset into pandas data frame.

[139]:
```
df = pd.read_csv("/content/drive/MyDrive/dsbda_datasets/StudentsPerformance.
↪csv")
```

[140]:
```
df
```

[140]:
```
    gender race/ethnicity parental level of education         lunch  \
0   female        group B           bachelor's degree      standard
1   female        group C                some college      standard
2   female        group B             master's degree      standard
3     male        group A          associate's degree  free/reduced
4     male        group C                some college      standard
..     …            …                         …             …
995 female        group E             master's degree      standard
996   male        group C                 high school  free/reduced
997 female        group C                 high school  free/reduced
998 female        group D                some college      standard
999 female        group D                some college  free/reduced

   test preparation course  math score  reading score  writing score
0                     none        72.0           72.0             74
1                completed        69.0           90.0             88
2                     none        90.0           95.0             93
3                     none        47.0           57.0             44
4                     none        76.0           78.0             75
..                      …           …             …             …
995              completed        88.0           99.0             95
996                   none        62.0           55.0             55
997              completed        59.0           71.0             65
998              completed        68.0           78.0             77
999                   none        77.0           86.0             86

[1000 rows x 8 columns]
```

3. Data Preprocessing

[141]:
```
# Checking for missing values

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

[141]:
```
gender                         0
race/ethnicity                 0
parental level of education    3
lunch                          0
test preparation course        0
math score                     1
reading score                  2
writing score                  0
dtype: int64
```

[142]:
```python
# initial statistics

df.describe()
```

[142]:

	math score	reading score	writing score
count	999.000000	998.000000	1000.000000
mean	66.093093	69.178357	68.054000
std	15.170122	14.611940	15.195657
min	0.000000	17.000000	10.000000
25%	57.000000	59.000000	57.750000
50%	66.000000	70.000000	69.000000
75%	77.000000	79.000000	79.000000
max	100.000000	100.000000	100.000000

[143]:
```python
# first few rows of dataset

df.head()
```

[143]:

	gender	race/ethnicity	parental level of education	lunch	\
0	female	group B	bachelor's degree	standard	
1	female	group C	some college	standard	
2	female	group B	master's degree	standard	
3	male	group A	associate's degree	free/reduced	
4	male	group C	some college	standard	

	test preparation course	math score	reading score	writing score
0	none	72.0	72.0	74
1	completed	69.0	90.0	88
2	none	90.0	95.0	93
3	none	47.0	57.0	44
4	none	76.0	78.0	75

[144]:
```python
# last few rows of dataset

df.tail()
```

[144]:

	gender	race/ethnicity	parental level of education	lunch	\
995	female	group E	master's degree	standard	
996	male	group C	high school	free/reduced	
997	female	group C	high school	free/reduced	
998	female	group D	some college	standard	
999	female	group D	some college	free/reduced	

	test preparation course	math score	reading score	writing score
995	completed	88.0	99.0	95
996	none	62.0	55.0	55
997	completed	59.0	71.0	65

```
998               completed      68.0          78.0            77
999                    none      77.0          86.0            86
```

[145]:
```
# Get information about the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   gender                       1000 non-null   object
 1   race/ethnicity               1000 non-null   object
 2   parental level of education  997 non-null    object
 3   lunch                        1000 non-null   object
 4   test preparation course      1000 non-null   object
 5   math score                   999 non-null    float64
 6   reading score                998 non-null    float64
 7   writing score                1000 non-null   int64
dtypes: float64(2), int64(1), object(5)
memory usage: 62.6+ KB
```

[146]:
```
# Dimensions of dataframe

df.shape
```

[146]:
```
(1000, 8)
```

[147]:
```
# Variable Descriptions

variable_descriptions = {
    'gender': 'Gender of the person',
    'race/ethnicity': 'Caste of the person',
    # Add more variable descriptions as needed
}

for column in df.columns:
  print(column +" : "+ variable_descriptions.get(column, 'No description␣
 ↪available'))
```

```
gender : Gender of the person
race/ethnicity : Caste of the person
parental level of education : No description available
lunch : No description available
test preparation course : No description available
math score : No description available
reading score : No description available
```

```
writing score : No description available
```

4. Turn categorical variables into quantitative variables in Python.

[148]:
```python
categorical_variables = ['gender']
```

[149]:
```python
# This is just another type of encoding (cat to quant conversion) No need to␣
↪look

# from sklearn.preprocessing import LabelEncoder
# encoded = LabelEncoder().fit_transform(df[categorical_variables[0]])
# encoded
```

[150]:
```python
encoded_df = pd.get_dummies(df,columns = categorical_variables) # one-hot␣
↪encoding
```

[151]:
```python
encoded_df.head()
```

[151]:
```
  race/ethnicity parental level of education         lunch  \
0        group B           bachelor's degree      standard
1        group C                some college      standard
2        group B             master's degree      standard
3        group A          associate's degree  free/reduced
4        group C                some college      standard

  test preparation course  math score  reading score  writing score  \
0                    none        72.0           72.0             74
1               completed        69.0           90.0             88
2                    none        90.0           95.0             93
3                    none        47.0           57.0             44
4                    none        76.0           78.0             75

   gender_female  gender_male
0              1            0
1              1            0
2              1            0
3              0            1
4              0            1
```

3 Assignment 3 (Data Wrangling)

1. Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use the following techniques to deal with them.

a) Delete rows or column
b) replace missing values with mean
c) replace missing values with mode
d) replace missing values with median

a. Import all the required Python Libraries.

[152]:
```
import pandas as pd
```

b. Load the dataset

[153]:
```
df = pd.read_csv("/content/drive/MyDrive/dsbda_datasets/StudentsPerformance.
↪csv")
```

c. Check for missing values

[154]:
```
df.isnull().sum()
```

[154]:
```
gender                         0
race/ethnicity                 0
parental level of education    3
lunch                          0
test preparation course        0
math score                     1
reading score                  2
writing score                  0
dtype: int64
```

d. Delete rows or column having null values

[155]:
```
df.shape
```

[155]:
```
(1000, 8)
```

[156]:
```
# df = df.dropna() # drop rows where null value found
# df = df.dropna(axis = 1) # drop columns where null value found
```

[157]:
```
df.shape
```

[157]:
```
(1000, 8)
```

[158]:
```
df.isnull().sum()
```

[158]:
```
gender                         0
race/ethnicity                 0
parental level of education    3
lunch                          0
test preparation course        0
math score                     1
reading score                  2
writing score                  0
dtype: int64
```

e. Replace missing values with mean

[159]:
```
# df.fillna(df.mean())
```

f. Replace missing values with mode

[160]:
```
# df.fillna(df.mode().iloc[0])
```

g. Replace missing values with median

[161]:
```
df.fillna(df.median(numeric_only=True))
```

[161]:
```
     gender race/ethnicity parental level of education         lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male        group A          associate's degree  free/reduced
4      male        group C                some college      standard
..      …            …                        …            …
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none        72.0           72.0             74
1                 completed        69.0           90.0             88
2                      none        90.0           95.0             93
3                      none        47.0           57.0             44
4                      none        76.0           78.0             75
..                       …           …             …             …
995               completed        88.0           99.0             95
996                    none        62.0           55.0             55
997               completed        59.0           71.0             65
998               completed        68.0           78.0             77
999                    none        77.0           86.0             86

[1000 rows x 8 columns]
```

[162]:
```
df
```

[162]:
```
     gender race/ethnicity parental level of education         lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male        group A          associate's degree  free/reduced
4      male        group C                some college      standard
..      …            …                        …            …
995  female        group E             master's degree      standard
```

```
996    male      group C          high school  free/reduced
997  female      group C          high school  free/reduced
998  female      group D         some college      standard
999  female      group D         some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none        72.0           72.0             74
1                 completed        69.0           90.0             88
2                      none        90.0           95.0             93
3                      none        47.0           57.0             44
4                      none        76.0           78.0             75
..                      …          …              …             …
995               completed        88.0           99.0             95
996                    none        62.0           55.0             55
997               completed        59.0           71.0             65
998               completed        68.0           78.0             77
999                    none        77.0           86.0             86

[1000 rows x 8 columns]
```

2. Apply data transformations on at least one of the variables. The purpose of this transformation should be one of the following reasons: to change the scale for better understanding of the variable, to convert a non-linear relation into a linear one, or to decrease the skewness and convert the distribution into a normal distribution.

a. Import all the required Python Libraries.

[163]:
```python
import numpy as np
```

[164]:
```python
df
```

[164]:
```
     gender race/ethnicity parental level of education         lunch  \
0    female        group B           bachelor's degree      standard
1    female        group C                some college      standard
2    female        group B             master's degree      standard
3      male        group A          associate's degree  free/reduced
4      male        group C                some college      standard
..      …            …                        …           …
995  female        group E             master's degree      standard
996    male        group C                 high school  free/reduced
997  female        group C                 high school  free/reduced
998  female        group D                some college      standard
999  female        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none        72.0           72.0             74
1                 completed        69.0           90.0             88
2                      none        90.0           95.0             93
```

```
3                      none       47.0          57.0            44
4                      none       76.0          78.0            75
..                        …          …             …             …
995               completed       88.0          99.0            95
996                    none       62.0          55.0            55
997               completed       59.0          71.0            65
998               completed       68.0          78.0            77
999                    none       77.0          86.0            86

[1000 rows x 8 columns]
```

b. Apply a data transformation on a variable

[165]:
```
transformed_variable = np.log(df['reading score'])  # apply logarithmic␣
↪transformation (natural log)
```

[166]:
```
transformed_variable
```

[166]:
```
0      4.276666
1      4.499810
2      4.553877
3      4.043051
4      4.356709
         …
995    4.595120
996    4.007333
997    4.262680
998    4.356709
999    4.454347
Name: reading score, Length: 1000, dtype: float64
```

4 Assignment 4 (Data Wrangling)

1. Import required libraries

[167]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

2. Load the dataset

[168]:
```
df = sns.load_dataset("titanic")
```

3. Scan all numeric variables in dataset

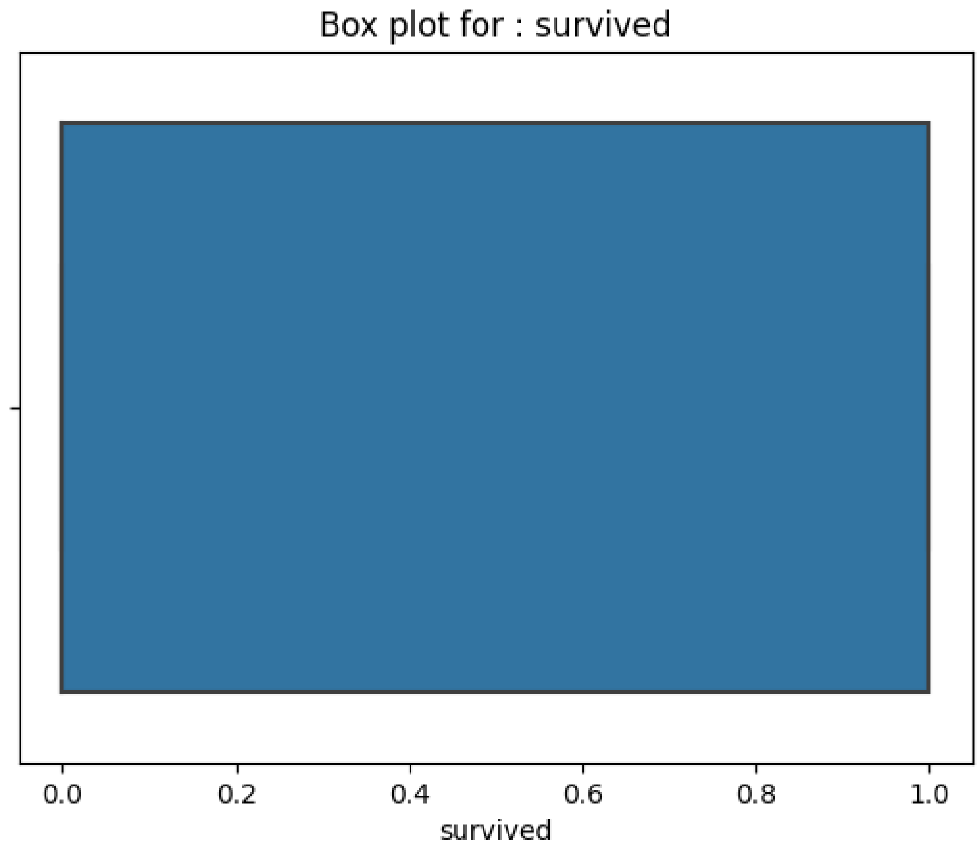
[169]:
```
numeric_variables = df.select_dtypes(include=['int','float']).columns.tolist()
```

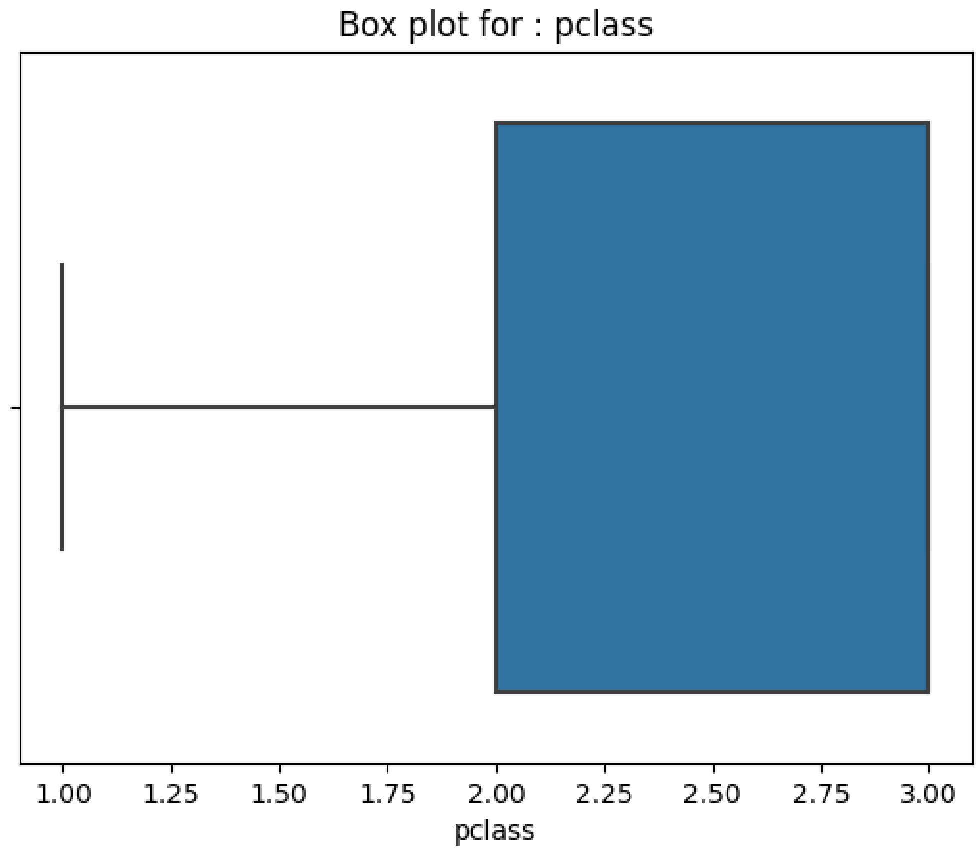
[170]:
```
numeric_variables
```

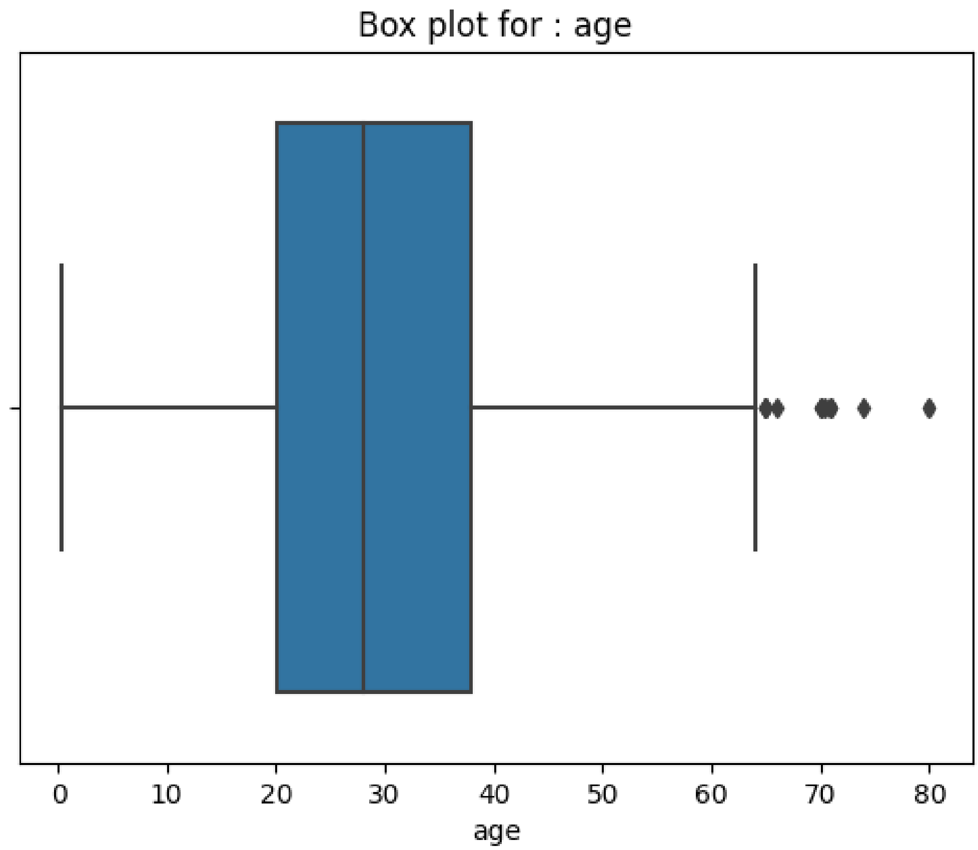
[170]: ['survived', 'pclass', 'age', 'sibsp', 'parch', 'fare']

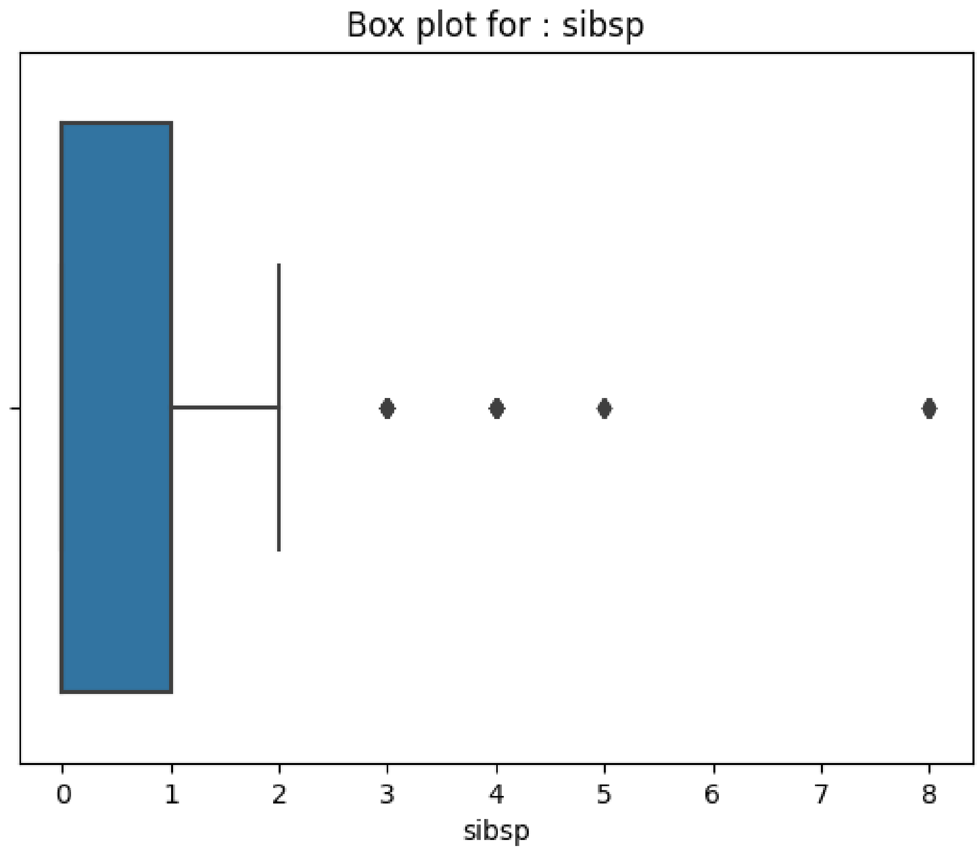
4. Scan numeric variables for outliers using box plots

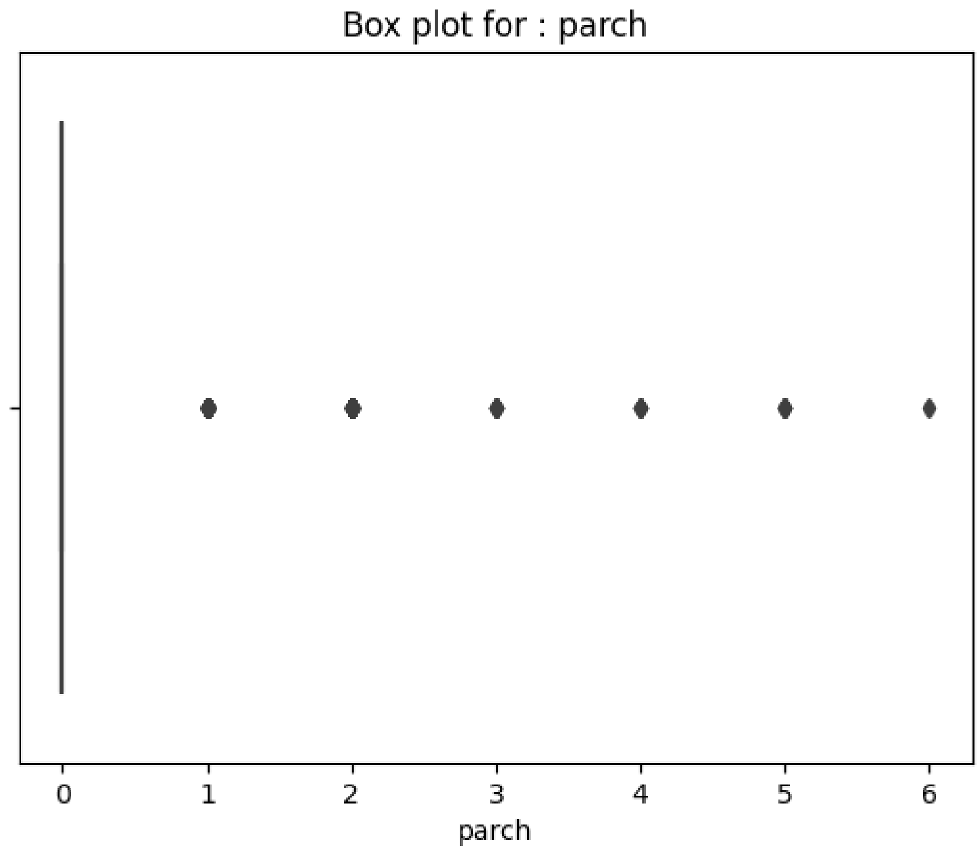
[171]:
```
for variable in numeric_variables:
  # Create a box plot for variable
  sns.boxplot(x=df[variable])
  plt.title(f'Box plot for : {variable}')
  plt.show()
  print("\n")

  # Identify outliers based on the box plot
  q1 = df[variable].quantile(0.25)
  q3 = df[variable].quantile(0.75)
  iqr = q3 - q1
  lower_bound = q1 - 1.5 * iqr
  upper_bound = q3 + 1.5 * iqr

  # Handle outliers using different techniques
  # a) Min Max Normalization
  min_value = df[variable].min()
  max_value = df[variable].max()
  df[variable+'_minmax'] = (df[variable] - min_value) / (max_value - min_value)

  # b) Z score Normalization
  mean = df[variable].mean()
  std_dev = df[variable].std()
  df[variable+'_zscore'] = (df[variable] - mean) / std_dev

  # c) Remove Outliers
  df[variable+'_no_outliers'] = df[variable][((df[variable] > lower_bound) &␣
↪(df[variable] < upper_bound))]
```

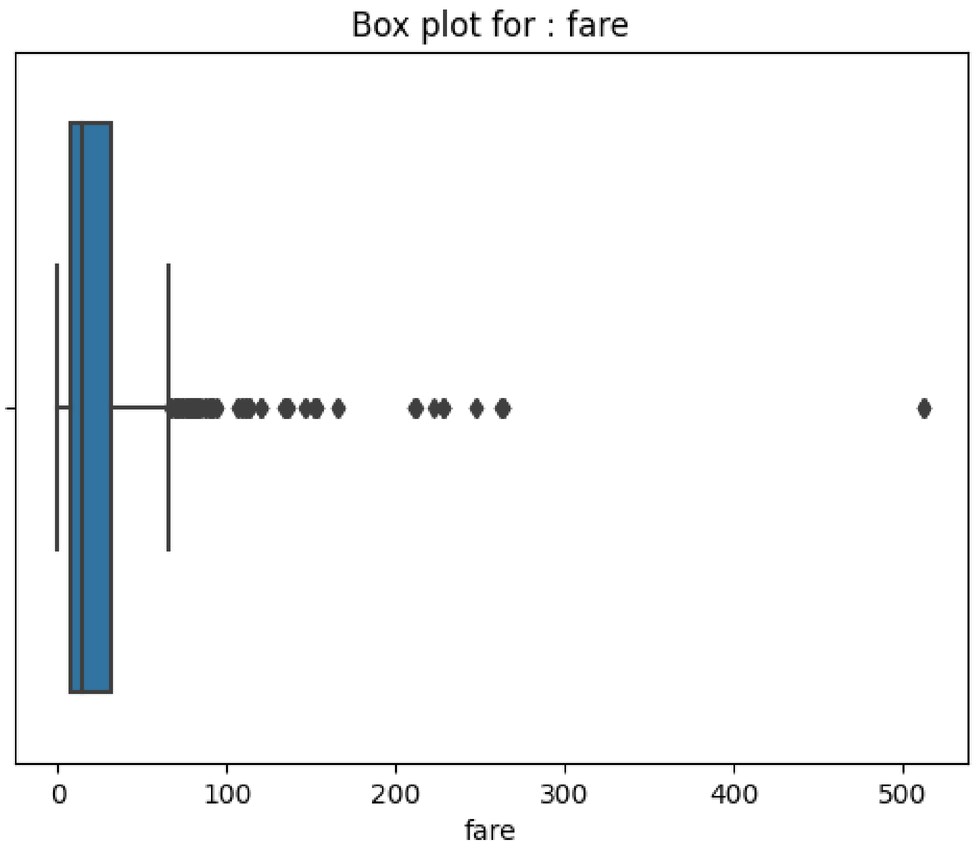

Box plot for : survived
0.0
0.2
0.4
0.6
0.8
1.0
survived


Box plot for : pclass
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
pclass


Box plot for : age
0
10
20
30
40
50
60
70
80
age


Box plot for : sibsp
0
1
2
3
4
5
6
7
8
sibsp


Box plot for : parch
0
1
2
3
4
5
6
parch



[172]:
```
df
```

[172]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        …       …       …     …      …      …        …        …       …
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  …  age_no_outliers sibsp_minmax sibsp_zscore sibsp_no_outliers  \
0      man  …             22.0        0.125     0.432550               1.0
```

```
1    woman  …            38.0        0.125     0.432550               1.0
2    woman  …            26.0        0.000    -0.474279               0.0
3    woman  …            35.0        0.125     0.432550               1.0
4      man  …            35.0        0.000    -0.474279               0.0
..     …  …             …           …          …                …
886    man  …            27.0        0.000    -0.474279               0.0
887  woman  …            19.0        0.000    -0.474279               0.0
888  woman  …             NaN        0.125     0.432550               1.0
889    man  …            26.0        0.000    -0.474279               0.0
890    man  …            32.0        0.000    -0.474279               0.0

     parch_minmax  parch_zscore  parch_no_outliers  fare_minmax  fare_zscore  \
0        0.000000     -0.473408                NaN     0.014151    -0.502163
1        0.000000     -0.473408                NaN     0.139136     0.786404
2        0.000000     -0.473408                NaN     0.015469    -0.488580
3        0.000000     -0.473408                NaN     0.103644     0.420494
4        0.000000     -0.473408                NaN     0.015713    -0.486064
..            …            …                 …          …          …
886      0.000000     -0.473408                NaN     0.025374    -0.386454
887      0.000000     -0.473408                NaN     0.058556    -0.044356
888      0.333333      2.007806                NaN     0.045771    -0.176164
889      0.000000     -0.473408                NaN     0.058556    -0.044356
890      0.000000     -0.473408                NaN     0.015127    -0.492101

     fare_no_outliers
0               7.250
1                 NaN
2               7.925
3              53.100
4               8.050
..                …
886            13.000
887            30.000
888            23.450
889            30.000
890             7.750

[891 rows x 33 columns]
```

5 Assignment 5 (Statistics)

1. Import all required libraries

[173]:
```python
import pandas as pd
```

[174]:
```python
df = pd.read_csv("/content/drive/MyDrive/dsbda_datasets/nba.csv")
```

[175]:
```
df
```

[175]:
```
              Name            Team  Number Position   Age Height  Weight  \
0    Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
1      Jae Crowder  Boston Celtics    99.0       SF  25.0    6-6   235.0
2     John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0
3      R.J. Hunter  Boston Celtics    28.0       SG  22.0    6-5   185.0
4    Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0
..             …             …     …        …     …      …       …
453   Shelvin Mack       Utah Jazz     8.0       PG  26.0    6-3   203.0
454      Raul Neto       Utah Jazz    25.0       PG  24.0    6-1   179.0
455   Tibor Pleiss       Utah Jazz    21.0        C  26.0    7-3   256.0
456    Jeff Withey       Utah Jazz    24.0        C  26.0    7-0   231.0
457            NaN             NaN     NaN      NaN   NaN    NaN     NaN

               College     Salary
0                Texas  7730337.0
1            Marquette  6796117.0
2    Boston University        NaN
3        Georgia State  1148640.0
4                  NaN  5000000.0
..                 …          …
453             Butler  2433333.0
454                NaN   900000.0
455                NaN  2900000.0
456             Kansas   947276.0
457                NaN        NaN

[458 rows x 9 columns]
```

2. Setting Variables

[176]:
```
categorical_variable = 'Team'
quantitative_variable = 'Salary'
```

3. Grouping The data frame

[177]:
```
df_grouped = df.groupby(by=categorical_variable)
```

[178]:
```
df_grouped.describe()
```

[178]:
```
                       Number                                                  \
                        count       mean        std  min    25%   50%    75%
Team
Atlanta Hawks            15.0  19.000000  11.476684  0.0  11.50  17.0  25.50
Boston Celtics           15.0  31.866667  30.300558  0.0   9.50  28.0  42.50
Brooklyn Nets            15.0  18.266667  14.104035  0.0   8.00  15.0  27.00
Charlotte Hornets        15.0  17.133333  16.672761  0.0   4.00  12.0  27.50
```

Chicago Bulls	15.0	19.200000	17.193022	0.0	5.50	16.0	28.00
Cleveland Cavaliers	15.0	14.466667	13.809245	0.0	4.50	12.0	21.50
Dallas Mavericks	15.0	20.000000	16.252472	1.0	6.00	21.0	30.50
Denver Nuggets	15.0	15.266667	19.655849	0.0	4.00	9.0	18.00
Detroit Pistons	15.0	17.266667	15.303906	0.0	5.50	13.0	23.50
Golden State Warriors	15.0	20.866667	11.413442	4.0	11.50	20.0	30.50
Houston Rockets	15.0	14.666667	12.505237	0.0	5.50	12.0	25.50
Indiana Pacers	15.0	18.933333	15.988686	0.0	4.00	13.0	30.50
Los Angeles Clippers	15.0	19.533333	13.125040	3.0	8.50	19.0	31.00
Los Angeles Lakers	15.0	16.066667	15.285225	0.0	3.50	9.0	26.00
Memphis Grizzlies	18.0	15.555556	14.030313	0.0	5.50	10.5	21.25
Miami Heat	15.0	10.466667	10.377632	0.0	3.50	8.0	13.00
Milwaukee Bucks	16.0	20.000000	17.485232	3.0	10.50	17.5	21.25
Minnesota Timberwolves	14.0	19.571429	21.964007	1.0	8.25	13.0	21.75
New Orleans Pelicans	19.0	17.000000	14.011900	0.0	4.50	15.0	27.50
New York Knicks	16.0	13.250000	12.964053	1.0	4.75	8.5	17.25
Oklahoma City Thunder	15.0	14.000000	12.130246	0.0	4.50	11.0	21.50
Orlando Magic	14.0	16.428571	16.411601	0.0	5.50	10.5	20.75
Philadelphia 76ers	15.0	18.066667	14.660280	0.0	6.00	12.0	32.00
Phoenix Suns	15.0	15.466667	10.405127	1.0	7.00	15.0	22.00
Portland Trail Blazers	15.0	16.000000	13.711309	0.0	4.50	11.0	23.50
Sacramento Kings	15.0	16.933333	12.002777	0.0	7.50	15.0	25.50
San Antonio Spurs	15.0	17.933333	11.067757	1.0	10.50	17.0	23.50
Toronto Raptors	15.0	22.466667	25.856380	1.0	5.50	10.0	27.50
Utah Jazz	15.0	17.866667	11.432202	2.0	9.00	20.0	24.50
Washington Wizards	15.0	17.600000	22.610996	1.0	5.50	12.0	19.00

		Age		…	Weight		Salary	\
	max	count	mean	…	75%	max	count	
Team				…				
Atlanta Hawks	43.0	15.0	28.200000	…	242.50	260.0	15.0	
Boston Celtics	99.0	15.0	24.733333	…	236.50	260.0	14.0	
Brooklyn Nets	44.0	15.0	25.600000	…	220.50	275.0	15.0	
Charlotte Hornets	50.0	15.0	26.133333	…	240.00	289.0	15.0	
Chicago Bulls	55.0	15.0	27.400000	…	231.00	275.0	15.0	
Cleveland Cavaliers	52.0	15.0	29.533333	…	250.50	275.0	14.0	
Dallas Mavericks	50.0	15.0	29.733333	…	245.00	275.0	15.0	
Denver Nuggets	77.0	15.0	25.733333	…	226.50	280.0	14.0	
Detroit Pistons	50.0	15.0	26.200000	…	242.50	279.0	15.0	
Golden State Warriors	40.0	15.0	27.666667	…	247.50	273.0	15.0	
Houston Rockets	35.0	15.0	26.866667	…	237.50	265.0	15.0	
Indiana Pacers	44.0	15.0	26.400000	…	246.50	255.0	15.0	
Los Angeles Clippers	45.0	15.0	29.466667	…	242.50	265.0	15.0	
Los Angeles Lakers	50.0	15.0	27.533333	…	250.00	270.0	15.0	
Memphis Grizzlies	50.0	18.0	28.388889	…	236.75	270.0	14.0	
Miami Heat	40.0	15.0	28.933333	…	237.50	265.0	13.0	
Milwaukee Bucks	77.0	16.0	24.562500	…	246.75	265.0	16.0	

```
Minnesota Timberwolves  88.0  14.0  26.357143  …  240.75  307.0   13.0
New Orleans Pelicans    44.0  19.0  26.894737  …  235.00  270.0   19.0
New York Knicks         43.0  16.0  27.000000  …  240.00  278.0   16.0
Oklahoma City Thunder   35.0  15.0  27.066667  …  247.50  255.0   15.0
Orlando Magic           55.0  14.0  25.071429  …  238.75  260.0   14.0
Philadelphia 76ers      42.0  15.0  24.600000  …  246.50  275.0   14.0
Phoenix Suns            35.0  15.0  25.866667  …  241.00  260.0   15.0
Portland Trail Blazers  44.0  15.0  25.066667  …  240.00  265.0   15.0
Sacramento Kings        41.0  15.0  26.800000  …  239.00  270.0   15.0
San Antonio Spurs       40.0  15.0  31.600000  …  245.00  290.0   15.0
Toronto Raptors         92.0  15.0  26.133333  …  242.50  255.0   15.0
Utah Jazz               41.0  15.0  24.466667  …  232.50  265.0   15.0
Washington Wizards      90.0  15.0  27.866667  …  241.00  250.0   15.0

                                                                          \
                                mean           std        min         25%
Team
Atlanta Hawks           4.860197e+06  5.194508e+06   525093.0  1152260.00
Boston Celtics          4.181505e+06  3.146033e+06  1148640.0  1909560.00
Brooklyn Nets           3.501898e+06  5.317817e+06   134215.0   947276.00
Charlotte Hornets       5.222728e+06  4.538601e+06   189455.0  1543138.00
Chicago Bulls           5.785559e+06  6.251088e+06   525093.0  1203290.50
Cleveland Cavaliers     7.642049e+06  7.730329e+06   111196.0  1179457.00
Dallas Mavericks        4.746582e+06  5.030279e+06   525093.0  1185783.00
Denver Nuggets          4.294424e+06  4.320214e+06   258489.0  1615789.75
Detroit Pistons         4.477884e+06  4.668478e+06   111444.0  1711452.50
Golden State Warriors   5.924600e+06  5.664282e+06   289755.0  1201462.00
Houston Rockets         5.018868e+06  6.414749e+06   200600.0   973638.00
Indiana Pacers          4.450122e+06  4.584514e+06   211744.0  1053513.00
Los Angeles Clippers    6.323643e+06  7.600225e+06   111444.0  1024164.00
Los Angeles Lakers      4.784695e+06  6.835688e+06   525093.0   896167.50
Memphis Grizzlies       5.467920e+06  5.201676e+06   700902.0  1274280.00
Miami Heat              6.347359e+06  7.848628e+06   261894.0   947276.00
Milwaukee Bucks         4.350220e+06  4.875071e+06   295327.0  1483589.00
Minnesota Timberwolves  4.593054e+06  4.139625e+06   947276.0  1474440.00
New Orleans Pelicans    4.355304e+06  4.537874e+06    55722.0   981348.50
New York Knicks         4.581494e+06  5.952487e+06    30888.0   921721.75
Oklahoma City Thunder   6.251020e+06  6.632400e+06   222888.0  1742280.00
Orlando Magic           4.297248e+06  3.068412e+06   845059.0  2311302.00
Philadelphia 76ers      2.213778e+06  1.900402e+06   525093.0   947276.00
Phoenix Suns            4.229676e+06  5.022561e+06    55722.0   964312.00
Portland Trail Blazers  3.220121e+06  2.392741e+06   525093.0  1181398.00
Sacramento Kings        4.778911e+06  4.701792e+06   525093.0   998384.50
San Antonio Spurs       5.629516e+06  6.396804e+06   200600.0  1045078.00
Toronto Raptors         4.741174e+06  4.195943e+06   245177.0  1683000.00
Utah Jazz               4.204006e+06  4.467878e+06   900000.0  1262160.00
Washington Wizards      5.088576e+06  4.869388e+06   200600.0  1510421.00
```

```
                              50%          75%         max
Team
Atlanta Hawks           2854940.0   6873239.50  18671659.0
Boston Celtics          3021242.5   6347087.75  12000000.0
Brooklyn Nets           1335480.0   2512675.00  19689000.0
Charlotte Hornets       4204200.0   6665702.00  13500000.0
Chicago Bulls           2380440.0   7974380.00  20093064.0
Cleveland Cavaliers     4975000.0  12942843.75  22970500.0
Dallas Mavericks        3950313.0   5289487.00  16407500.0
Denver Nuggets          2907000.0   4142083.25  14000000.0
Detroit Pistons         2891760.0   5635000.00  16000000.0
Golden State Warriors   3815000.0  11540621.00  15501000.0
Houston Rockets         2288205.0   7339758.00  22359364.0
Indiana Pacers          4000000.0   5697112.50  17120106.0
Los Angeles Clippers    3110796.0   8367500.00  21468695.0
Los Angeles Lakers      1724250.0   5161144.50  25000000.0
Memphis Grizzlies       4544009.5   8116000.00  19688000.0
Miami Heat              2481720.0  10151612.00  22192730.0
Milwaukee Bucks         2254167.0   5514330.00  16407500.0
Minnesota Timberwolves  2148360.0   5758680.00  12700000.0
New Orleans Pelicans    2850000.0   7785365.00  15514031.0
New York Knicks         2225421.0   4949493.00  22875000.0
Oklahoma City Thunder   3344000.0   8694215.00  20158622.0
Orlando Magic           3956580.0   5144390.00  11250000.0
Philadelphia 76ers      1037084.5   3310710.00   6500000.0
Phoenix Suns            2041080.0   5500000.00  13500000.0
Portland Trail Blazers  2854940.0   4626143.50   8042895.0
Sacramento Kings        3156600.0   6880303.00  15851950.0
San Antonio Spurs       2814000.0   8750000.00  19689000.0
Toronto Raptors         2900000.0   6634337.50  13600000.0
Utah Jazz               2433333.0   4276360.00  15409570.0
Washington Wizards      4000000.0   6847337.00  15851950.0

[30 rows x 32 columns]
```

[179]:
```
df_grouped[quantitative_variable].describe()
```

[179]:
```
                        count          mean           std        min  \
Team
Atlanta Hawks            15.0  4.860197e+06  5.194508e+06   525093.0
Boston Celtics           14.0  4.181505e+06  3.146033e+06  1148640.0
Brooklyn Nets            15.0  3.501898e+06  5.317817e+06   134215.0
Charlotte Hornets        15.0  5.222728e+06  4.538601e+06   189455.0
Chicago Bulls            15.0  5.785559e+06  6.251088e+06   525093.0
Cleveland Cavaliers      14.0  7.642049e+06  7.730329e+06   111196.0
```

Dallas Mavericks	15.0	4.746582e+06	5.030279e+06	525093.0
Denver Nuggets	14.0	4.294424e+06	4.320214e+06	258489.0
Detroit Pistons	15.0	4.477884e+06	4.668478e+06	111444.0
Golden State Warriors	15.0	5.924600e+06	5.664282e+06	289755.0
Houston Rockets	15.0	5.018868e+06	6.414749e+06	200600.0
Indiana Pacers	15.0	4.450122e+06	4.584514e+06	211744.0
Los Angeles Clippers	15.0	6.323643e+06	7.600225e+06	111444.0
Los Angeles Lakers	15.0	4.784695e+06	6.835688e+06	525093.0
Memphis Grizzlies	14.0	5.467920e+06	5.201676e+06	700902.0
Miami Heat	13.0	6.347359e+06	7.848628e+06	261894.0
Milwaukee Bucks	16.0	4.350220e+06	4.875071e+06	295327.0
Minnesota Timberwolves	13.0	4.593054e+06	4.139625e+06	947276.0
New Orleans Pelicans	19.0	4.355304e+06	4.537874e+06	55722.0
New York Knicks	16.0	4.581494e+06	5.952487e+06	30888.0
Oklahoma City Thunder	15.0	6.251020e+06	6.632400e+06	222888.0
Orlando Magic	14.0	4.297248e+06	3.068412e+06	845059.0
Philadelphia 76ers	14.0	2.213778e+06	1.900402e+06	525093.0
Phoenix Suns	15.0	4.229676e+06	5.022561e+06	55722.0
Portland Trail Blazers	15.0	3.220121e+06	2.392741e+06	525093.0
Sacramento Kings	15.0	4.778911e+06	4.701792e+06	525093.0
San Antonio Spurs	15.0	5.629516e+06	6.396804e+06	200600.0
Toronto Raptors	15.0	4.741174e+06	4.195943e+06	245177.0
Utah Jazz	15.0	4.204006e+06	4.467878e+06	900000.0
Washington Wizards	15.0	5.088576e+06	4.869388e+06	200600.0

	25%	50%	75%	max
Team				
Atlanta Hawks	1152260.00	2854940.0	6873239.50	18671659.0
Boston Celtics	1909560.00	3021242.5	6347087.75	12000000.0
Brooklyn Nets	947276.00	1335480.0	2512675.00	19689000.0
Charlotte Hornets	1543138.00	4204200.0	6665702.00	13500000.0
Chicago Bulls	1203290.50	2380440.0	7974380.00	20093064.0
Cleveland Cavaliers	1179457.00	4975000.0	12942843.75	22970500.0
Dallas Mavericks	1185783.00	3950313.0	5289487.00	16407500.0
Denver Nuggets	1615789.75	2907000.0	4142083.25	14000000.0
Detroit Pistons	1711452.50	2891760.0	5635000.00	16000000.0
Golden State Warriors	1201462.00	3815000.0	11540621.00	15501000.0
Houston Rockets	973638.00	2288205.0	7339758.00	22359364.0
Indiana Pacers	1053513.00	4000000.0	5697112.50	17120106.0
Los Angeles Clippers	1024164.00	3110796.0	8367500.00	21468695.0
Los Angeles Lakers	896167.50	1724250.0	5161144.50	25000000.0
Memphis Grizzlies	1274280.00	4544009.5	8116000.00	19688000.0
Miami Heat	947276.00	2481720.0	10151612.00	22192730.0
Milwaukee Bucks	1483589.00	2254167.0	5514330.00	16407500.0
Minnesota Timberwolves	1474440.00	2148360.0	5758680.00	12700000.0
New Orleans Pelicans	981348.50	2850000.0	7785365.00	15514031.0
New York Knicks	921721.75	2225421.0	4949493.00	22875000.0

```
Oklahoma City Thunder   1742280.00  3344000.0   8694215.00  20158622.0
Orlando Magic           2311302.00  3956580.0   5144390.00  11250000.0
Philadelphia 76ers       947276.00  1037084.5   3310710.00   6500000.0
Phoenix Suns             964312.00  2041080.0   5500000.00  13500000.0
Portland Trail Blazers  1181398.00  2854940.0   4626143.50   8042895.0
Sacramento Kings         998384.50  3156600.0   6880303.00  15851950.0
San Antonio Spurs       1045078.00  2814000.0   8750000.00  19689000.0
Toronto Raptors         1683000.00  2900000.0   6634337.50  13600000.0
Utah Jazz               1262160.00  2433333.0   4276360.00  15409570.0
Washington Wizards      1510421.00  4000000.0   6847337.00  15851950.0
```

6 Assignment 6 (Statistics)

1. Import all required libraries

[180]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[181]:
```
df = pd.read_csv("/content/drive/MyDrive/dsbda_datasets/Iris.csv")
```

2. Grouping on the basis of species

[182]:
```
categorical_variable = "Species"
```

[183]:
```
df_grouped = df.groupby(by=categorical_variable)
```

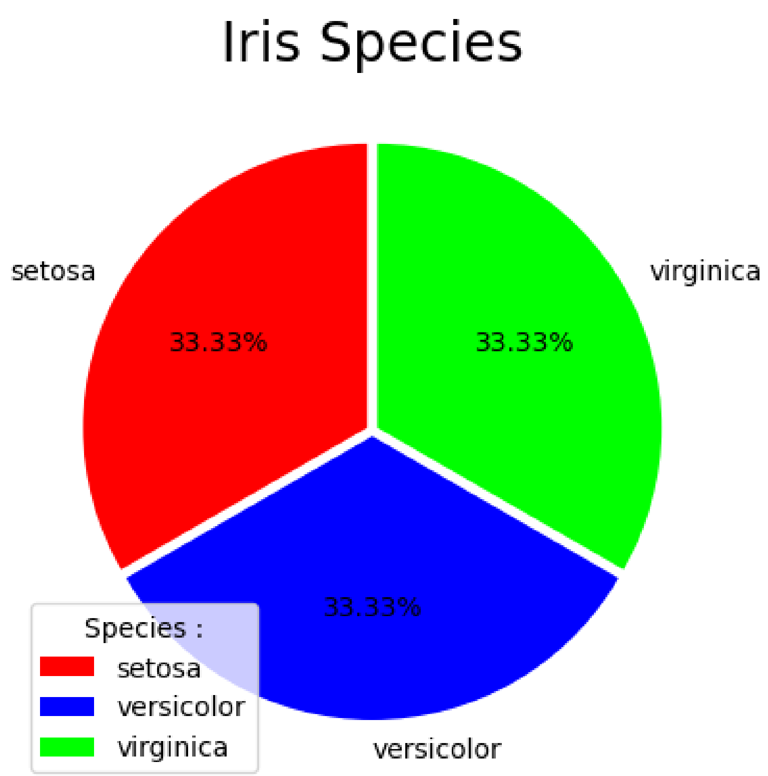
[184]:
```
df_grouped.describe()
```

[184]:
```
                   Id                                                      \
                count   mean       std    min     25%    50%     75%    max
Species
Iris-setosa      50.0   25.5  14.57738    1.0   13.25   25.5   37.75   50.0
Iris-versicolor  50.0   75.5  14.57738   51.0   63.25   75.5   87.75  100.0
Iris-virginica   50.0  125.5  14.57738  101.0  113.25  125.5  137.75  150.0

                SepalLengthCm         … PetalLengthCm      PetalWidthCm  \
                        count   mean  …           75%  max        count
Species                               …
Iris-setosa              50.0  5.006  …         1.575  1.9         50.0
Iris-versicolor          50.0  5.936  …         4.600  5.1         50.0
Iris-virginica           50.0  6.588  …         5.875  6.9         50.0


                  mean       std  min  25%  50%  75%  max
Species
Iris-setosa      0.244  0.107210  0.1  0.2  0.2  0.3  0.6
Iris-versicolor  1.326  0.197753  1.0  1.2  1.3  1.5  1.8
```

```
Iris-virginica   2.026  0.274650  1.4  1.8  2.0  2.3  2.5

[3 rows x 40 columns]
```

3. Plotting Pie chart on the basis of count of each species

[185]:
```
my_labels = ['setosa', 'versicolor', 'virginica']
size = df_grouped['Id'].count()
my_colors = ['red','b','#00FF00'] # accepts color name. color code and color␣
↪hex code
# 'r' - Red
# 'g' - Green
# 'b' - Blue
# 'c' - Cyan
# 'm' - Magenta
# 'y' - Yellow
# 'k' - Black
# 'w' - White
plt.pie(size, labels = my_labels, colors = my_colors ,shadow = False, autopct =␣
↪'%.2f%%',startangle=90,explode=[0.02,0.02,0.02])
plt.title('Iris Species', fontsize = 20)
plt.legend(title="Species :",loc="lower left")
plt.show()
```



Extra (Not required just look it once)

[186]:
```
df.mean(numeric_only=True)
```

[186]:
```
Id                 75.500000
SepalLengthCm       5.843333
SepalWidthCm        3.054000
PetalLengthCm       3.758667
PetalWidthCm        1.198667
dtype: float64
```

[187]:
```
col = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
for c in col:
   print(f'Column name : {c}')
   print(f'Column mean: {df[c].mean()}')
```

```
Column name : Id
Column mean: 75.5
Column name : SepalLengthCm
Column mean: 5.843333333333334
Column name : SepalWidthCm
Column mean: 3.0540000000000003
Column name : PetalLengthCm
Column mean: 3.758666666666666
Column name : PetalWidthCm
Column mean: 1.1986666666666668
```

7 Assignment 7 (Data Analytics I)

[188]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

[189]:
```
df = pd.read_csv("https://raw.githubusercontent.com/selva86/datasets/master/
 ↪BostonHousing.csv")
```

[190]:
```
# Split the dataset into features (X) and target variable (Y)
print(df.columns)
X = df['lstat'].values.reshape(-1, 1) # input features (Linear Regression)
# X = df.drop('medv',axis=1) # input features (Multiple Regression)
Y = df['medv'] # target variable
```

```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'b', 'lstat', 'medv'],
```

```
      dtype='object')
```

[191]:
```
# Split the data into training and test sets (25% Testing , 75% Training)
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.
 ↪25,random_state=5)
```

[192]:
```
# Create a linear regression model
model = LinearRegression()
```

[193]:
```
# Fit the model to the training data
model.fit(X_train,Y_train)
```

[193]:
```
LinearRegression()
```

[194]:
```
# Make predictions on the test data
Y_pred = model.predict(X_test)
```

[195]:
```
# Evaluate the model
rmse = np.sqrt(mean_squared_error(Y_test,Y_pred)) # or rmse =␣
 ↪mean_squared_error(Y_test,Y_pred,squared=False)
r2 = r2_score(Y_test,Y_pred)*100
```

[196]:
```
print("Root Mean Squared Error : ",rmse)
print("r2 Score (%): ",r2)
```

```
Root Mean Squared Error :  6.3239250854834745
r2 Score (%):  51.54907869513134
```

8 Assignment 8 (Data Analytics II)

[197]:
```
# Importing required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
 ↪recall_score

import warnings
warnings.filterwarnings("ignore") # used to ignore warnings
```

[198]:
```
# Importing dataset
df = pd.read_csv("https://raw.githubusercontent.com/shivang98/
 ↪Social-Network-ads-Boost/master/Social_Network_Ads.csv")
```

[199]:
```
df
```

[199]:
```
     User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
..        …       …   …              …        …
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

[400 rows x 5 columns]
```

[200]:
```python
# Separate the features and the target variable (Split the dataset)
X = df[['Age','EstimatedSalary']]
Y = df['Purchased']
```

[201]:
```python
# Splitting the dataset into training and testing sets (75% training, 25%␣
↪testing)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,␣
↪random_state = 0)
```

[202]:
```python
# Create an instance of LogisticRegression
model = LogisticRegression()
```

[203]:
```python
# Fit the model on the data
model.fit(X_train,Y_train)
```

[203]:
```
LogisticRegression()
```

[204]:
```python
# Perform predictions on the data
Y_pred = model.predict(X_test)
```

[205]:
```python
# Compute the confusion matrix
conf_matrix = confusion_matrix(Y_test,Y_pred)
```

[206]:
```python
TN = conf_matrix[0][0]
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
```

[207]:
```python
# Compute accuracy, error rate, precision, and recall

accuracy = accuracy_score(Y_test, Y_pred) #(TP + TN) / (TP+TN+FP+FN)
print("Accuracy : ",accuracy)
```

```
Accuracy :  0.68
```

```
[208]: precision = precision_score(Y_test,Y_pred,average="micro") #(TP) / (TP + FP)
print("Precision : ",precision)
```

```
Precision :  0.68
```

```
[209]: recall = recall_score(Y_test,Y_pred,average="micro") # TP / (TP+FN)
print("Recall : ",recall)
```

```
Recall :  0.68
```

```
[210]: error_rate = 1 - accuracy
print("Error rate : ", error_rate)
```

```
Error rate :  0.31999999999999995
```

9 Assignment 9 (Data Analytics III)

```
[211]: # Importing required libraries

import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score

import warnings
warnings.filterwarnings("ignore") # used to ignore warnings
```

```
[212]: # Importing dataset
df = pd.read_csv("https://raw.githubusercontent.com/venky14/
↪Machine-Learning-with-Iris-Dataset/master/Iris.csv")
```

```
[213]: df
```

```
[213]:       Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   …              …             …              …             …
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8
```

```
             Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

[214]:
```
X = df.drop('Species',axis=1)
Y = df['Species']
```

[215]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.
 ↪25,random_state=0)
```

[216]:
```
model = GaussianNB()
```

[217]:
```
model.fit(X_train,Y_train)
```

[217]:
```
GaussianNB()
```

[218]:
```
Y_pred = model.predict(X_test)
```

[219]:
```
conf_matrix = confusion_matrix(Y_test,Y_pred)
```

[220]:
```
TN = conf_matrix[0][0]
FP = conf_matrix[0][1]
FN = conf_matrix[1][0]
TP = conf_matrix[1][1]
```

[221]:
```
accuracy = accuracy_score(Y_test,Y_pred) # (TN + TP) / (TP+TN+FP+FN)
print("Accuracy : ",accuracy)
```

```
Accuracy :  1.0
```

[222]:
```
precision = precision_score(Y_test,Y_pred,average="micro") # TP / (TP + FP) #␣
 ↪why average = "micro" dont know just solved the error
print("Precision : ",precision)
```

```
Precision :  1.0
```

[223]:
```
recall = recall_score(Y_test,Y_pred,average="micro") # TP / (TP + FN)
print("Recall : ",recall)
```

```
Recall :  1.0
```

[224]:
```
error_rate = 1 - accuracy
print("Error Rate : ",error_rate)
```

```
Error Rate :  0.0
```

10 Assignment 10 (Text Analysis)

[225]:
```
import numpy as np

import nltk
# nltk.download(['punkt','averaged_perceptron_tagger','stopwords','wordnet'])
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
```

[226]:
```
# Sample Document
document = "The quick brown fox jumps over the lazy dog."
document_2 = "My dog is not lazy."
```

[227]:
```
# Tokenization
tokens = word_tokenize(document)
print("Tokens : ",tokens)
```

```
Tokens :  ['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy',
'dog', '.']
```

[228]:
```
# POS(Part-of-speech) Tagging
pos_tags = pos_tag(tokens)
print("POS Tags : ",pos_tags)
```

```
POS Tags :  [('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'),
('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN'),
('.', '.')]
```

[229]:
```
# Stop Words Removal
stop_words = set(stopwords.words("english"))
filtered_tokens = [token for token in tokens if token.lower() not in stop_words]
print("Filtered Tokens : ",filtered_tokens)
```

```
Filtered Tokens :  ['quick', 'brown', 'fox', 'jumps', 'lazy', 'dog', '.']
```

[230]:
```python
# Stemming
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
print("Stemmed Tokens : ",stemmed_tokens)
```

```
Stemmed Tokens :  ['quick', 'brown', 'fox', 'jump', 'lazi', 'dog', '.']
```

[231]:
```python
# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]
print("Lemmatized Tokens : ",lemmatized_tokens)
```

```
Lemmatized Tokens :  ['quick', 'brown', 'fox', 'jump', 'lazy', 'dog', '.']
```

[232]:
```python
# TF-IDF Representation

# Term Frequency
tokens = word_tokenize(document)
doc_dict = dict()
for token in tokens:
  if token in doc_dict.keys():
    doc_dict[token]+=1
  else:
    doc_dict[token] = 1
for key,value in doc_dict.items():
  doc_dict[key] = value/len(tokens)
print("TF of first document : ",doc_dict) # Term frequency of all words in␣
 ↪document

tokens = word_tokenize(document_2)
doc_dict_2 = dict()
for token in tokens:
  if token in doc_dict_2.keys():
    doc_dict_2[token]+=1
  else:
    doc_dict_2[token] = 1
for key,value in doc_dict_2.items():
  doc_dict_2[key] = value/len(tokens)
print("TF of second document : ",doc_dict_2) # Term frequency of all words in␣
 ↪document 2

# Inverse Document Frequency
total_no_of_docs = 2;
final_dict = dict()
for key in doc_dict.keys():
  if key in final_dict.keys():
    final_dict[key] += 1
```

```
  else:
    final_dict[key] = 1;
for key in doc_dict_2.keys():
  if key in final_dict.keys():
    final_dict[key] += 1
  else:
    final_dict[key] = 1;
for key,value in final_dict.items():
  final_dict[key] = np.log(total_no_of_docs/value)
print("IDF : ",final_dict)
```

```
TF of first document :  {'The': 0.1, 'quick': 0.1, 'brown': 0.1, 'fox': 0.1,
'jumps': 0.1, 'over': 0.1, 'the': 0.1, 'lazy': 0.1, 'dog': 0.1, '.': 0.1}
TF of second document :  {'My': 0.16666666666666666, 'dog': 0.16666666666666666,
'is': 0.16666666666666666, 'not': 0.16666666666666666, 'lazy':
0.16666666666666666, '.': 0.16666666666666666}
IDF :  {'The': 0.6931471805599453, 'quick': 0.6931471805599453, 'brown':
0.6931471805599453, 'fox': 0.6931471805599453, 'jumps': 0.6931471805599453,
'over': 0.6931471805599453, 'the': 0.6931471805599453, 'lazy': 0.0, 'dog': 0.0,
'.': 0.0, 'My': 0.6931471805599453, 'is': 0.6931471805599453, 'not':
0.6931471805599453}
```

11 Assignment ... (Data Visualization I)

[233]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[234]:
```
df = sns.load_dataset("titanic")
```

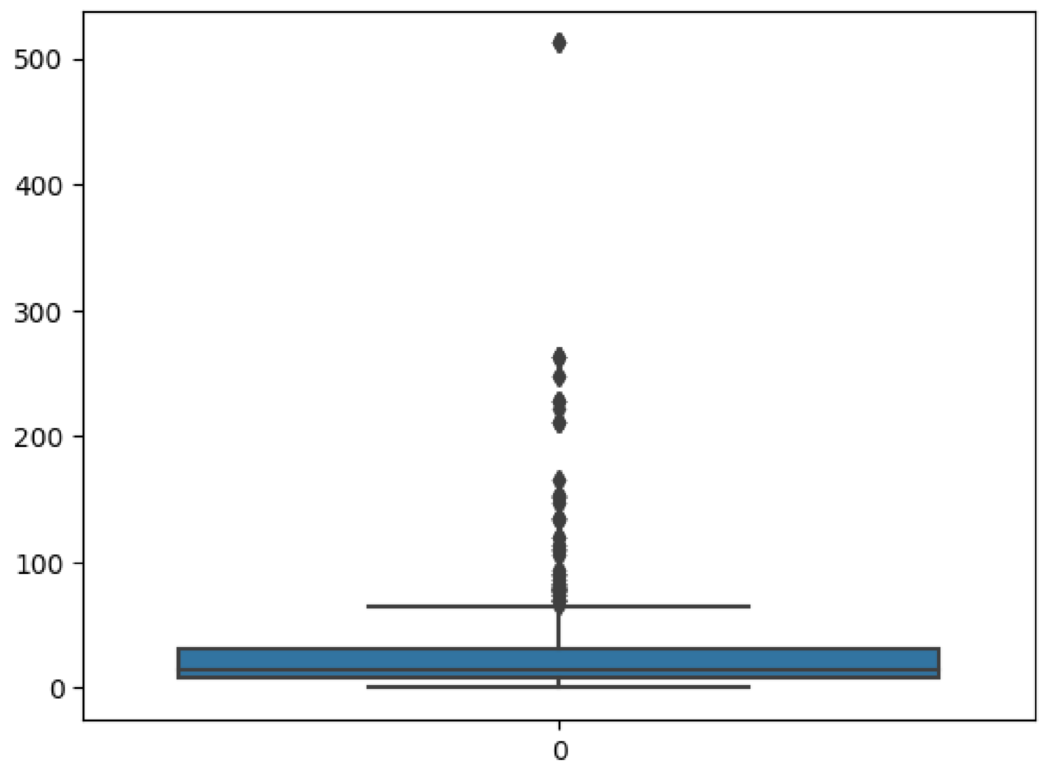
[235]:
```
df
```

[235]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        …       …     …     …      …      …        …      …       …
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
```

```
1    woman        False    C    Cherbourg   yes  False
2    woman        False  NaN  Southampton   yes   True
3    woman        False    C  Southampton   yes  False
4      man         True  NaN  Southampton    no   True
..     …            …    …           …    …    …
886    man         True  NaN  Southampton    no   True
887  woman        False    B  Southampton   yes   True
888  woman        False  NaN  Southampton    no  False
889    man         True    C    Cherbourg   yes   True
890    man         True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

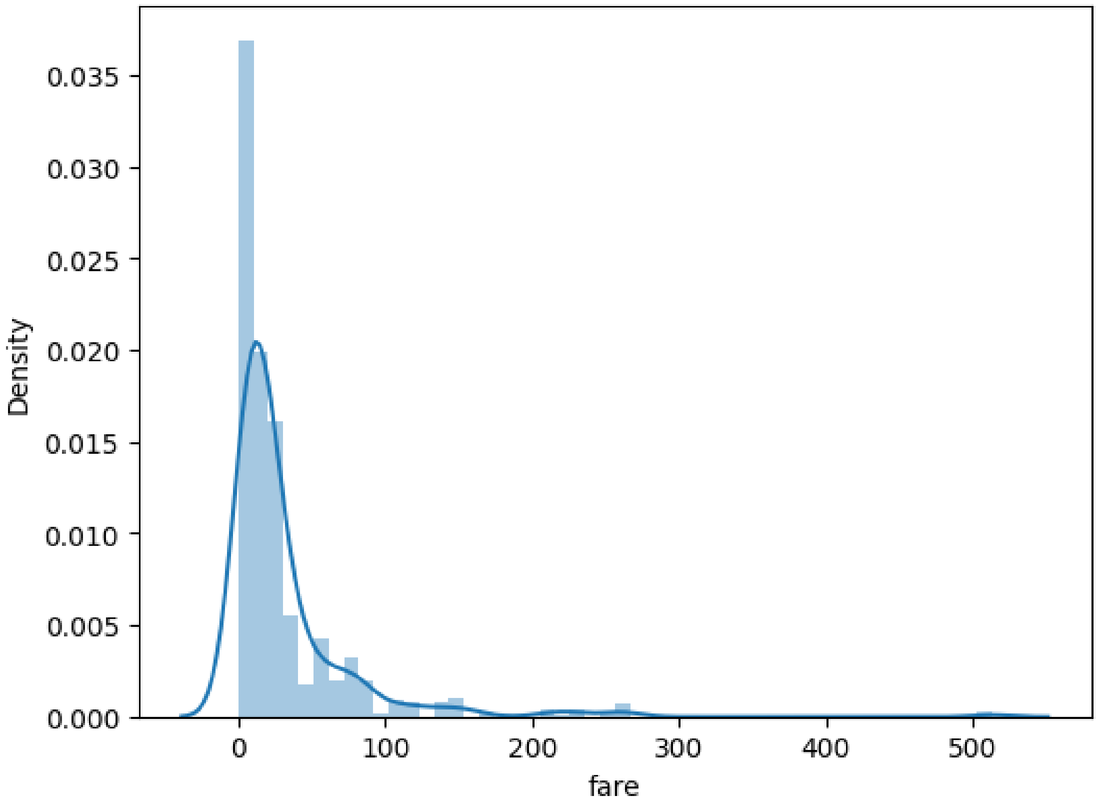
[236]:
```
sns.boxplot(df['fare'])
```

[236]: <Axes: >



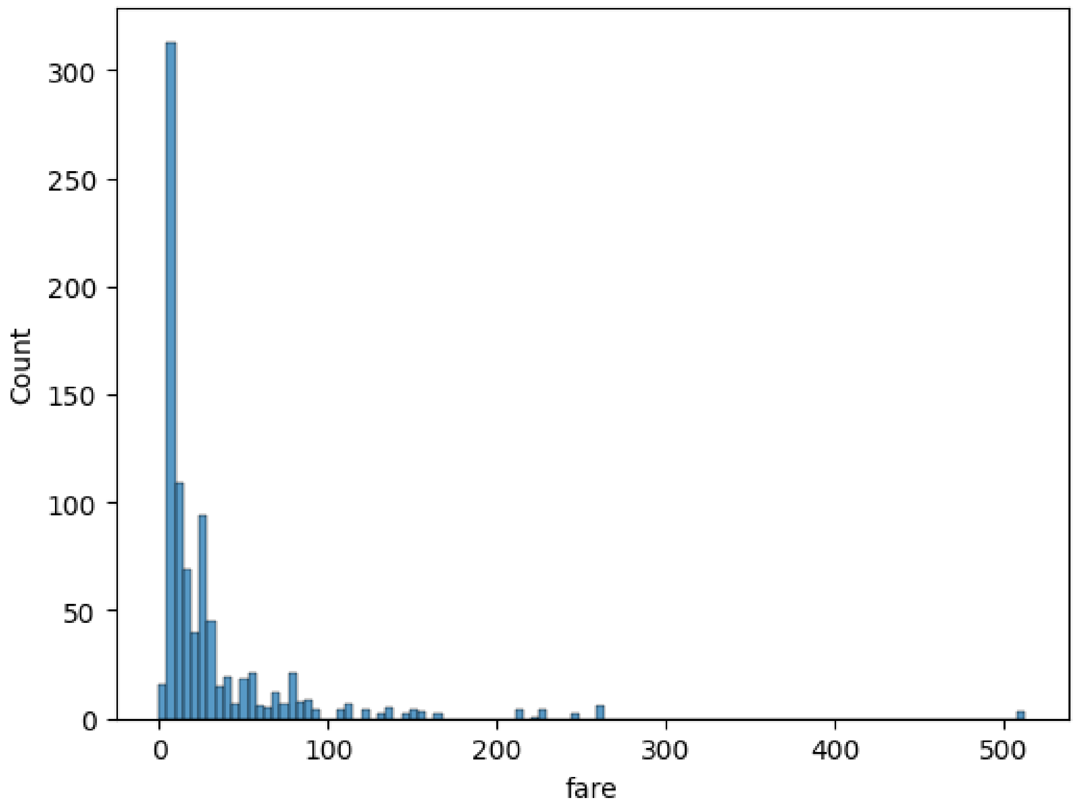
[237]:
```
sns.distplot(df['fare'])
```

[237]: <Axes: xlabel='fare', ylabel='Density'>



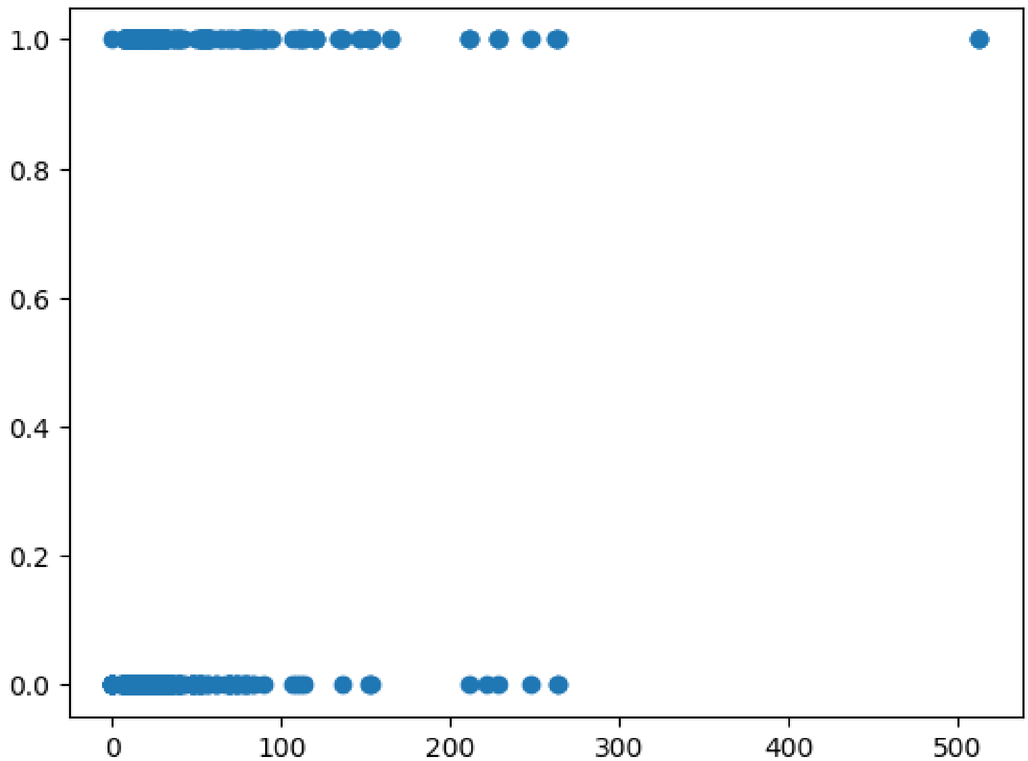
[238]:
```
sns.histplot(df['fare'])
```

[238]: <Axes: xlabel='fare', ylabel='Count'>



[239]:
```
plt.scatter(df['fare'],df['survived'])
```

[239]: <matplotlib.collections.PathCollection at 0x7f2ee9c37cd0>



12 Assignment 11 (Data Visualization II)

[240]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[241]:
```
# Load the titanic dataset
df = sns.load_dataset('titanic')
```

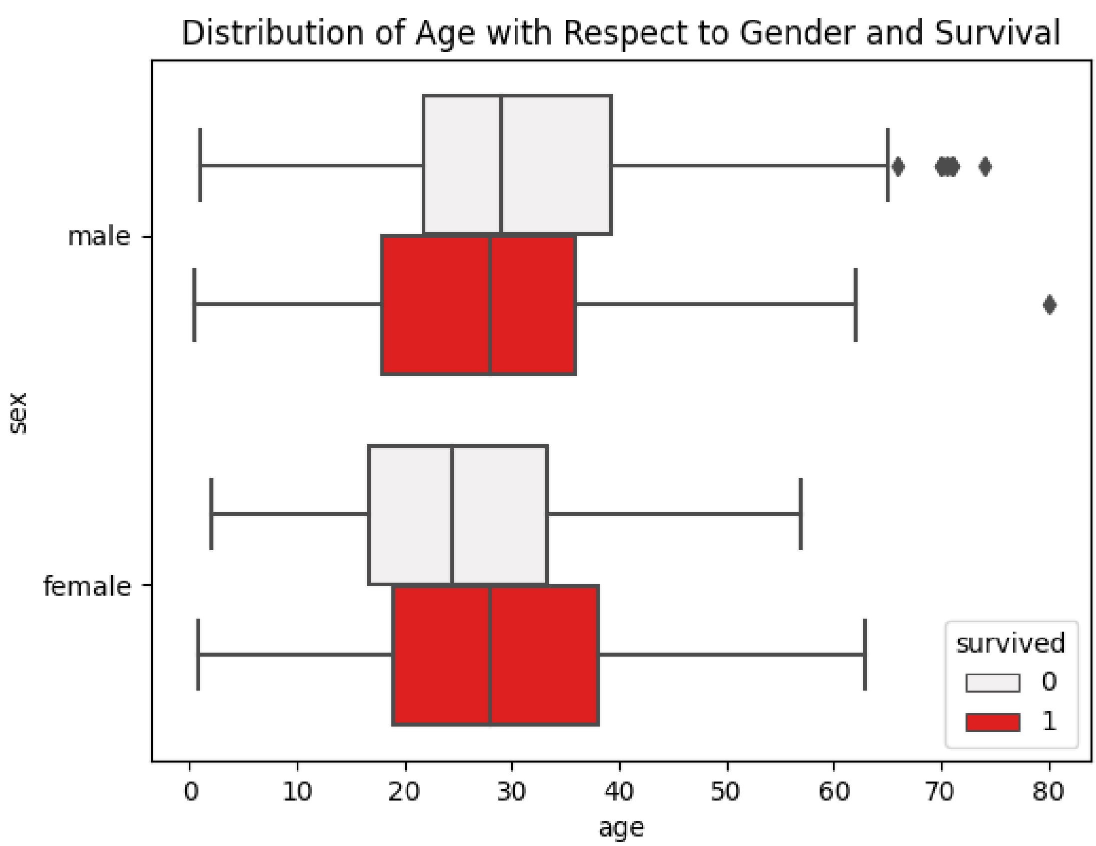
[242]:
```
df
```

[242]:
```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        …       …       …     …      …      …        …        …       …
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
```

```
890          0        3     male  32.0       0       0   7.7500         Q    Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     …          …   …          …    …     …
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

[243]:

```
sns.boxplot(x="age",y="sex",hue="survived",data=df,color="red")
plt.title('Distribution of Age with Respect to Gender and Survival')
plt.show()
```



13 Assignment 12 (Data Visualization III)

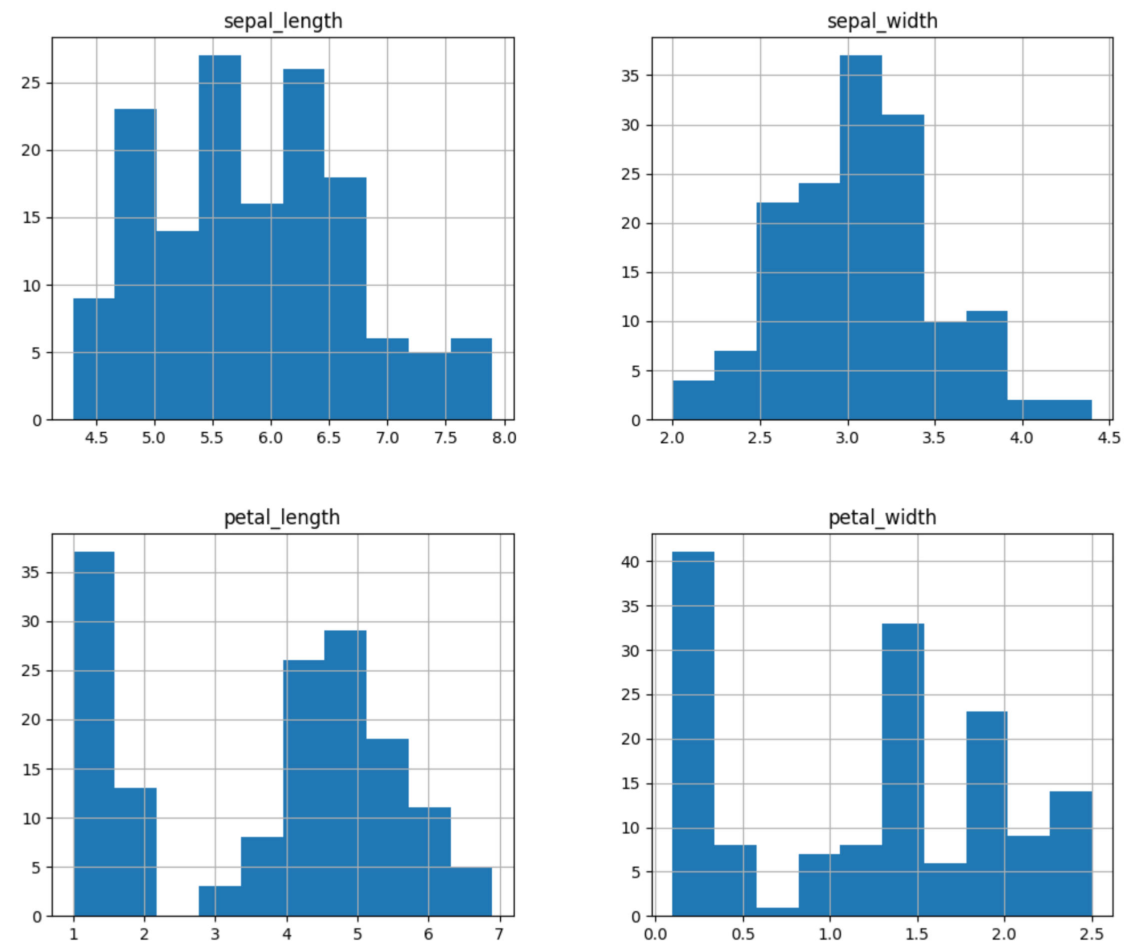
[244]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[245]:
```
df = sns.load_dataset("iris")
```

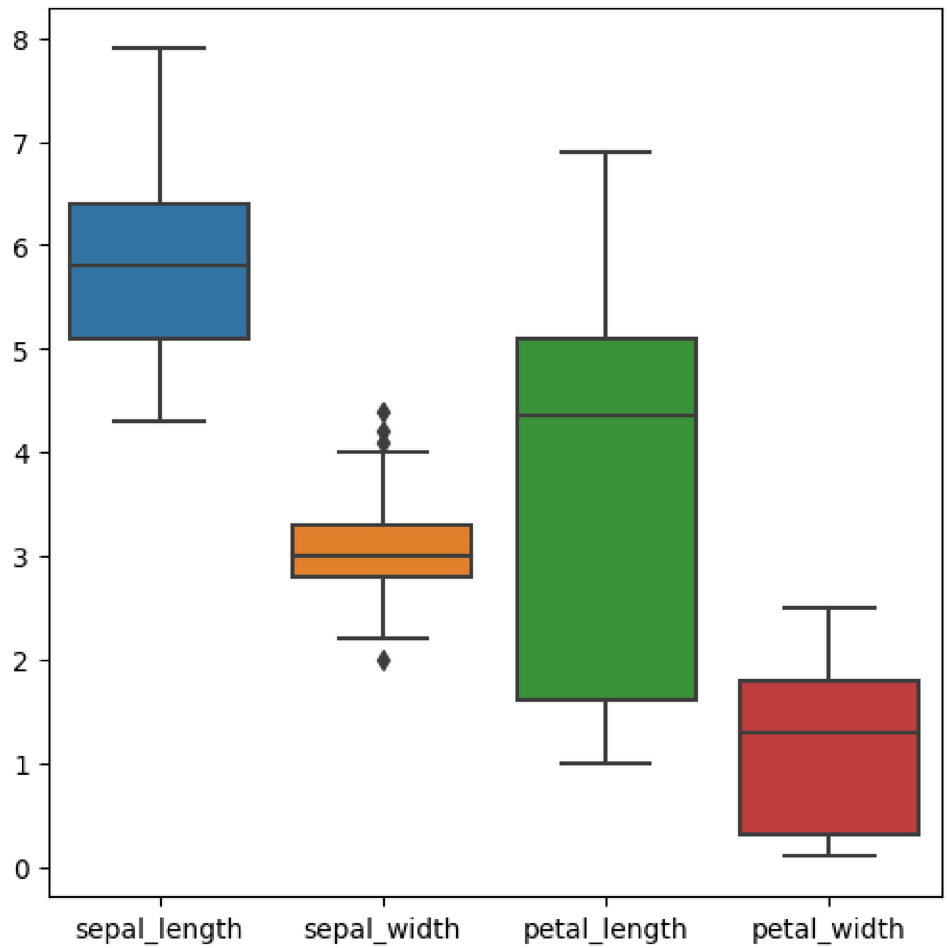
[246]:
```
# 1. List down the features and their types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[247]:
```
# 2. Create a histogram for each feature in the dataset to illustrate the
 ↪feature distributions.
df.hist(figsize=(12, 10))
# plt.title("BOX PLOT")
plt.show()
```



[248]:
```
# 3. Create a box plot for each feature
plt.figure(figsize=(6,6))
sns.boxplot(data=df)
plt.show()
```



[249]:
```
# 4. Compare distributions and identify outliers.
q1 = df['sepal_width'].quantile(0.25)
q3 = df['sepal_width'].quantile(0.75)
IQR = q3-q1
LB = q1-1.5*IQR
UB = q3+1.5*IQR
print(list(df[(df['sepal_width'] < LB)|(df['sepal_width'] > UB)].index)) #␣
↪Outliers
```

```
[15, 32, 33, 60]
```