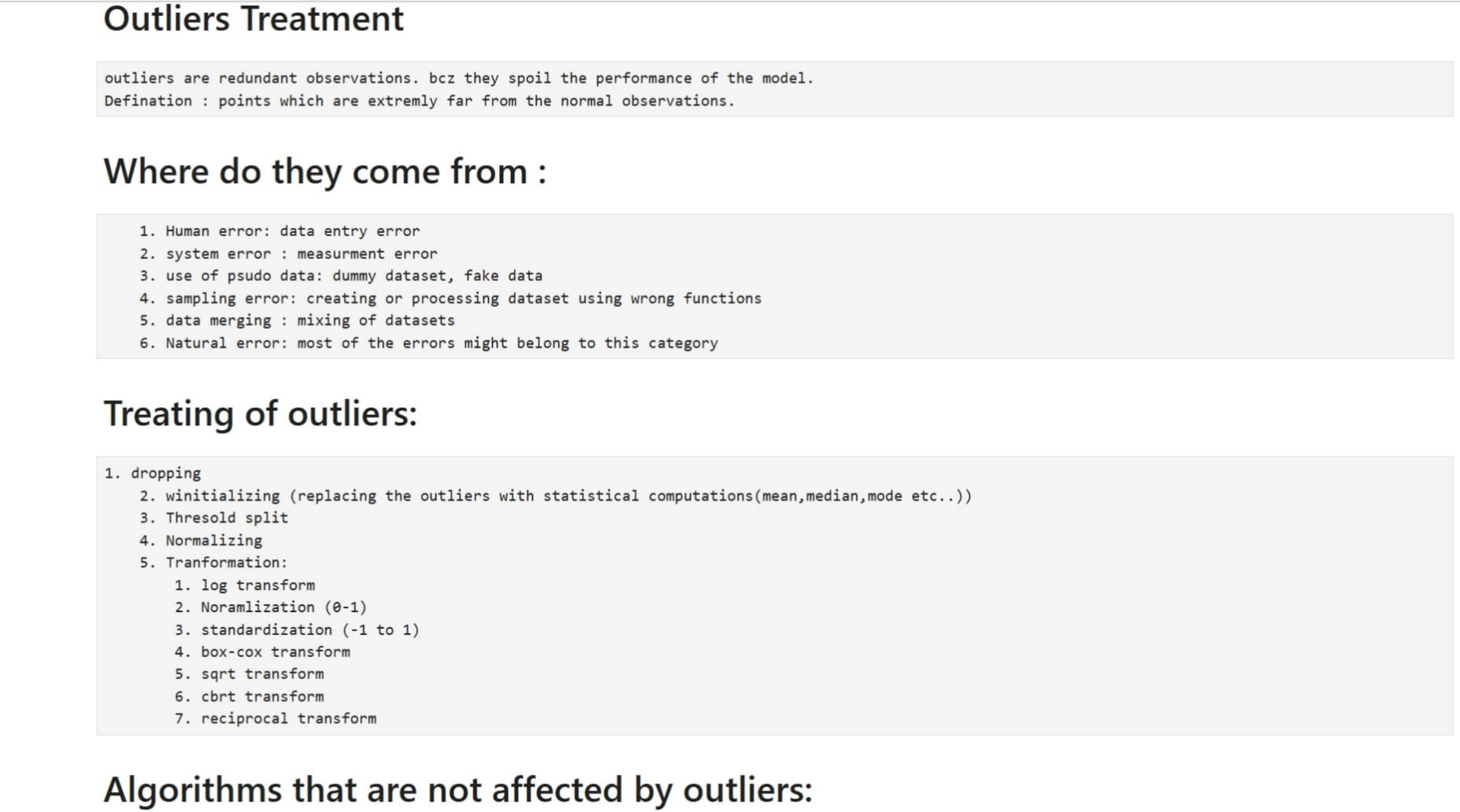


Outliers Treatment

```
outliers are redundant observations. bcz they spoil the performance of the model.
Defination : points which are extremly far from the normal observations.
```

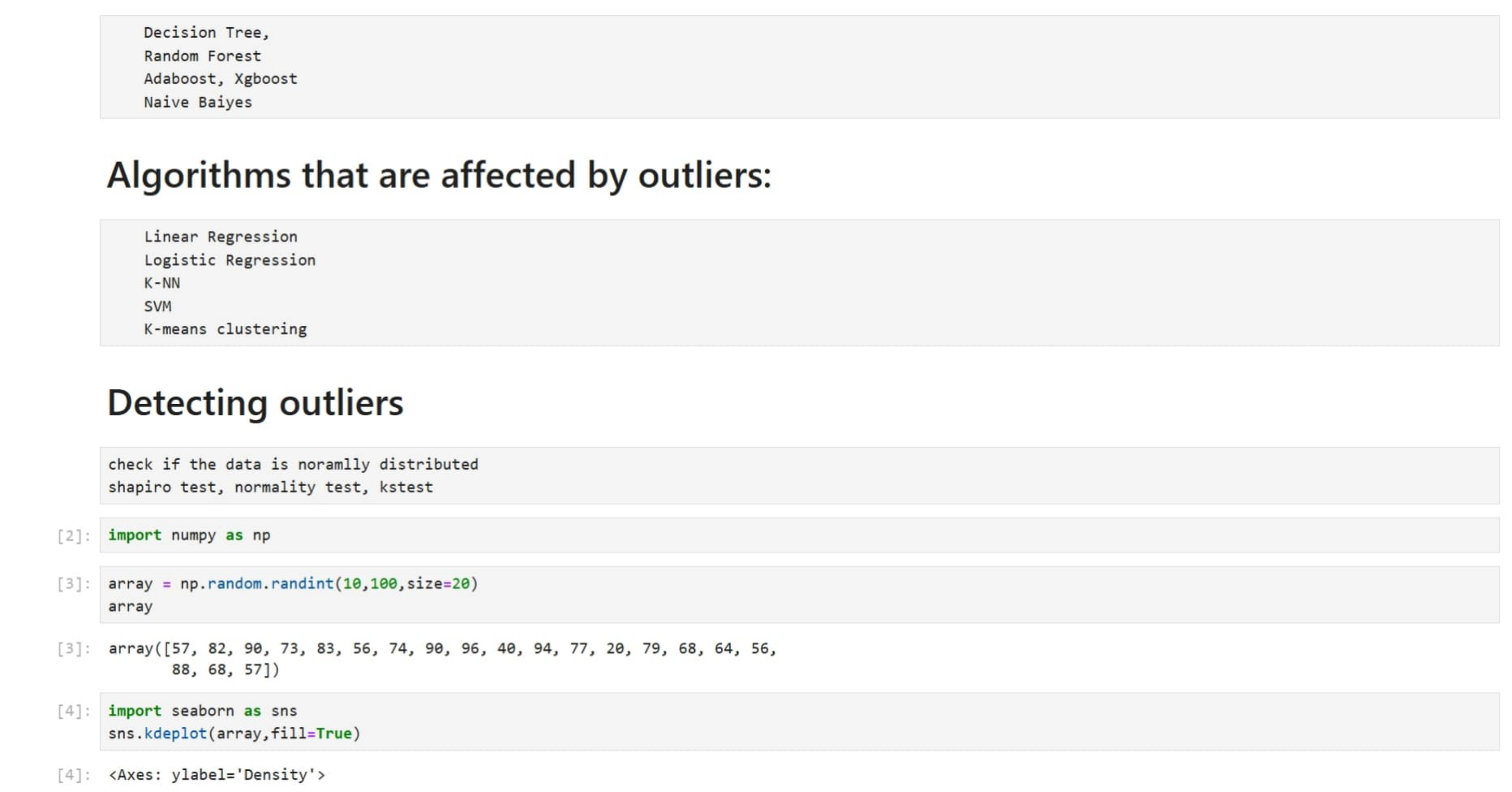
Where do they come from :

```
    1. Human error: data entry error
    2. system error : measurment error
    3. use of psudo data: dummy dataset, fake data
    4. sampling error: creating or processing dataset using wrong functions
    5. data merging : mixing of datasets
    6. Natural error: most of the errors might belong to this category
```

Treating of outliers:

```
1. dropping
    2. winitializing (replacing the outliers with statistical computations(mean,median,mode etc..))
    3. Thresold split
    4. Normalizing
    5. Tranformation:
        1. log transform
        2. Noramlization (0-1)
        3. standardization (-1 to 1)
        4. box-cox transform
        5. sqrt transform
        6. cbrt transform
        7. reciprocal transform
```

Algorithms that are not affected by outliers:



```
    Decision Tree,
    Random Forest
    Adaboost, Xgboost
    Naive Baiyes
```

Algorithms that are affected by outliers:

```
    Linear Regression
    Logistic Regression
    K-NN
    SVM
    K-means clustering
```

Detecting outliers

```
check if the data is noramlly distributed
shapiro test, normality test, kstest
```

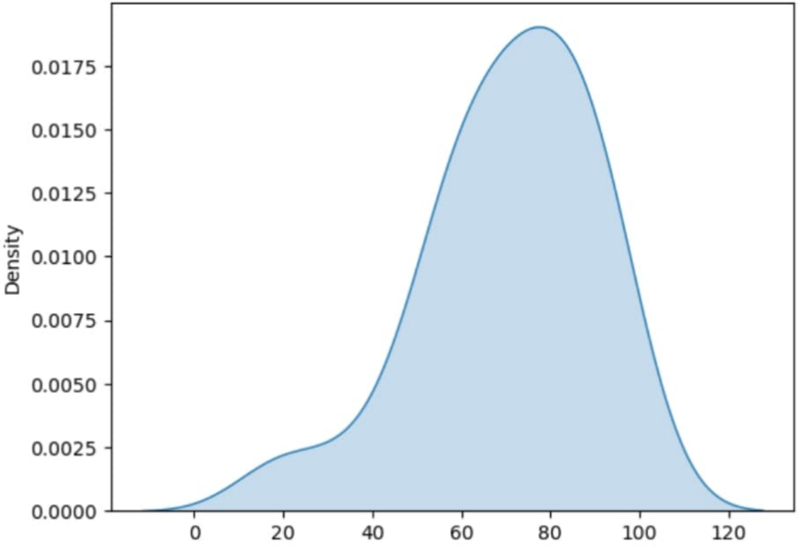
```
[2]: import numpy as np
```

```
[3]: array = np.random.randint(10,100,size=20)
     array
```

```
[3]: array([57, 82, 90, 73, 83, 56, 74, 90, 96, 40, 94, 77, 20, 79, 68, 64, 56,
            88, 68, 57])
```

```
[4]: import seaborn as sns
     sns.kdeplot(array,fill=True)
```

```
[4]: <Axes: ylabel='Density'>
```

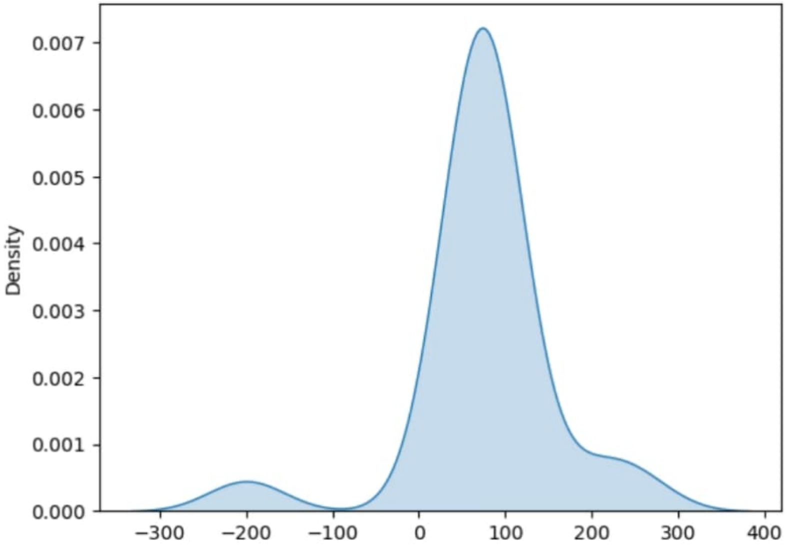


```
[8]: array2 = array.copy()
     array2[10]=200
     array2[12]=250
     array2[5]=-200
     array2
```

```
[8]: array([  57,   82,   90,   73,   83, -200,   74,   90,   96,   40,  200,
              77,  250,   79,   68,   64,   56,   88,   68,   57])
```

```
[9]: sns.kdeplot(array2,fill=True)
```

```
[9]: <Axes: ylabel='Density'>
```

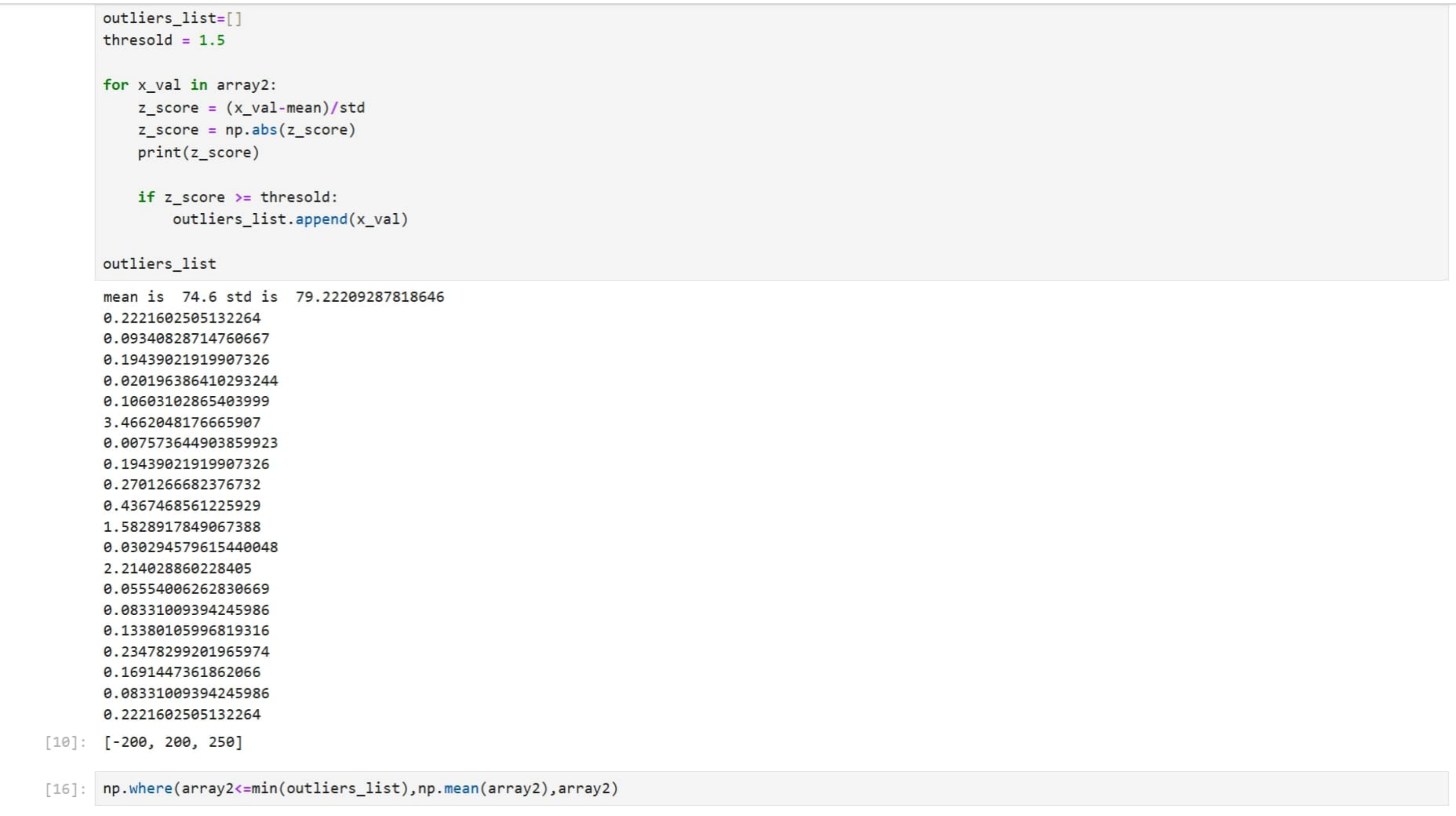


Z-score Normalization

```
z_score = (x_value -x_mean)/std

Idea is calculating z_score & setting up a thresold such that all the datapoints above the thresold or below the thresold are outliers & can be
eliminated.
```

[10]:

```
mean = np.mean(array2)
std = np.std(array2)
print('mean is ',mean, 'std is ',std)
```



```
outliers_list=[]
thresold = 1.5

for x_val in array2:
    z_score = (x_val-mean)/std
    z_score = np.abs(z_score)
    print(z_score)

    if z_score >= thresold:
        outliers_list.append(x_val)


outliers_list
```

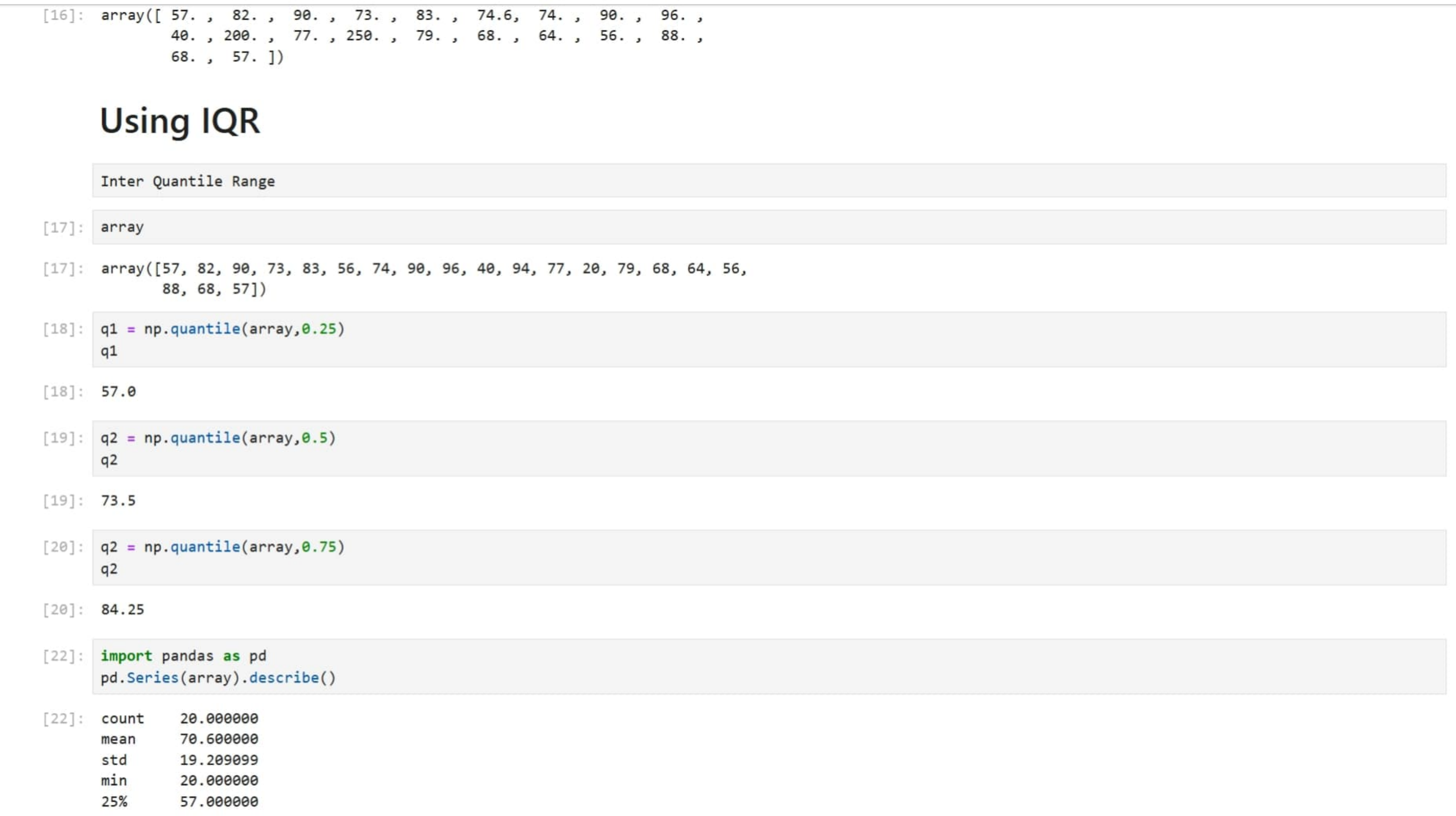
```
mean is  74.6 std is  79.22209287818646
0.2221602505132264
0.09340828714760667
0.19439021919907326
0.020196386410293244
0.10603102865403999
3.4662048176665907
0.007573644903859923
0.19439021919907326
0.2701266682376732
0.4367468561225929
1.5828917849067388
0.030294579615440048
2.214028860228405
0.05554006262830669
0.08331009394245986
0.13380105996819316
0.23478299201965974
0.1691447361862066
0.08331009394245986
0.2221602505132264
```

[10]:

```
[-200, 200, 250]
```

[16]:

```
np.where(array2<=min(outliers_list),np.mean(array2),array2)
```



```
[16]: array([ 57. ,  82. ,  90. ,  73. ,  83. ,  74.6,  74. ,  90. ,  96. ,
              40. , 200. ,  77. , 250. ,  79. ,  68. ,  64. ,  56. ,  88. ,
              68. ,  57. ])
```

Using IQR

```
Inter Quantile Range
```

```
[17]: array
```

```
[17]: array([57, 82, 90, 73, 83, 56, 74, 90, 96, 40, 94, 77, 20, 79, 68, 64, 56,
             88, 68, 57])
```

```
[18]: q1 = np.quantile(array,0.25)
      q1
```

```
[18]: 57.0
```

```
[19]: q2 = np.quantile(array,0.5)
      q2
```

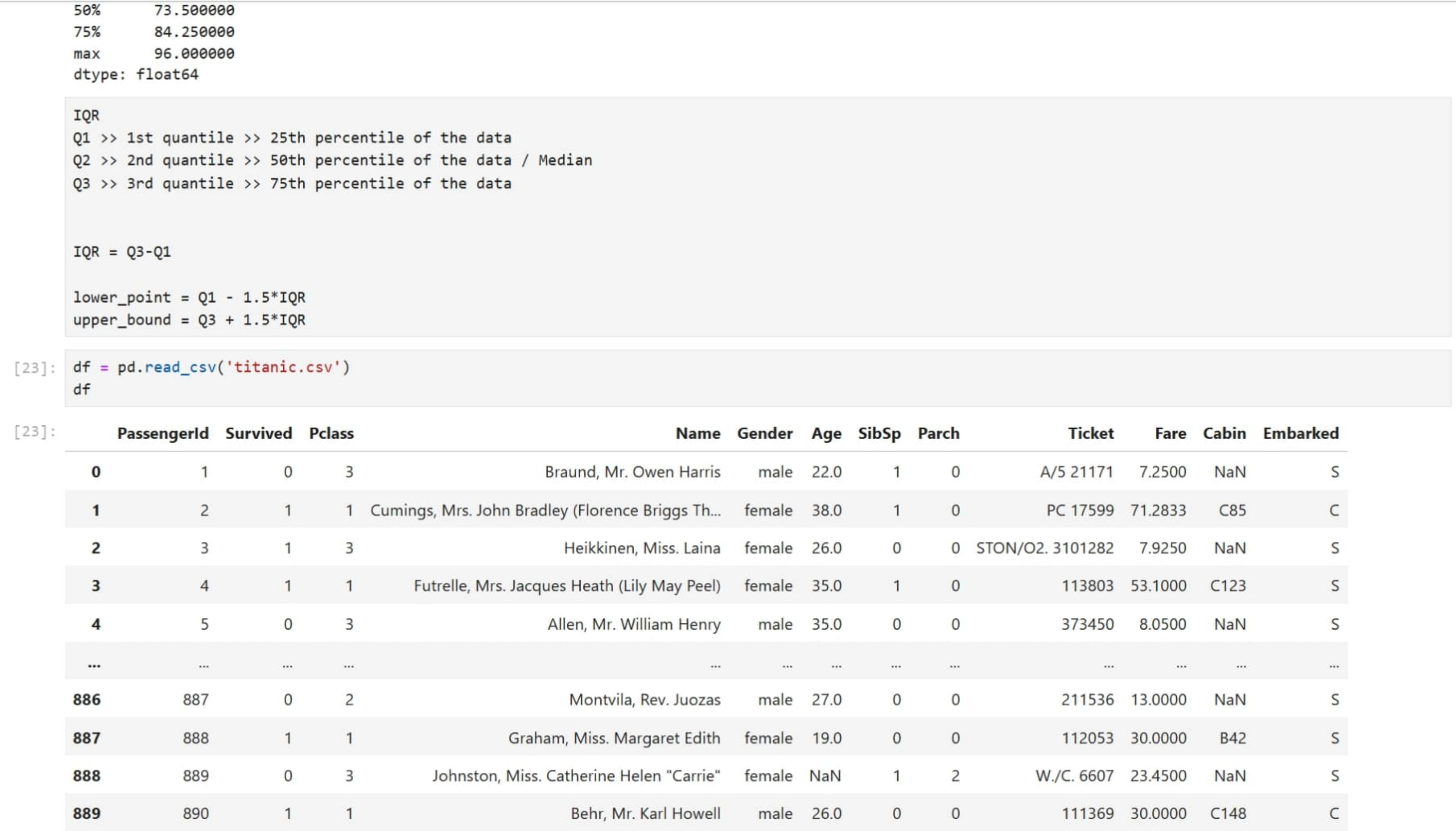
```
[19]: 73.5
```

```
[20]: q2 = np.quantile(array,0.75)
      q2
```

```
[20]: 84.25
```

```
[22]: import pandas as pd
      pd.Series(array).describe()
```

```
[22]: count    20.000000
      mean     70.600000
      std      19.209099
      min      20.000000
      25%      57.000000
```



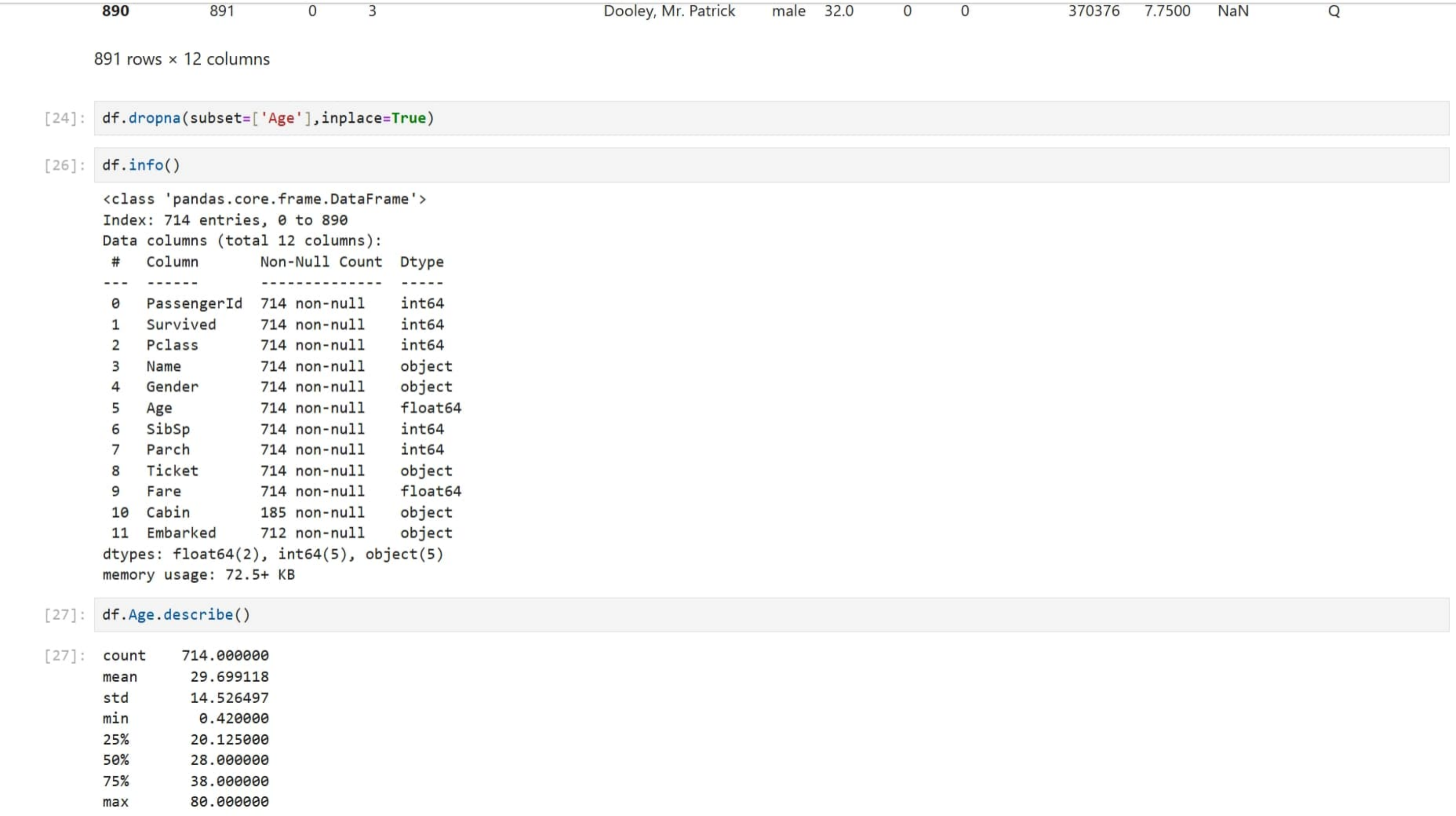
```
50%       73.500000
75%       84.250000
max       96.000000
dtype: float64
```

```
IQR
Q1 >> 1st quantile >> 25th percentile of the data
Q2 >> 2nd quantile >> 50th percentile of the data / Median
Q3 >> 3rd quantile >> 75th percentile of the data


IQR = Q3-Q1

lower_point = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR
```

```
[23]: df = pd.read_csv('titanic.csv')
      df
```

[23]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C



```
890    891    0    3    Dooley, Mr. Patrick    male    32.0    0    0    370376    7.7500    NaN    Q
```

891 rows × 12 columns

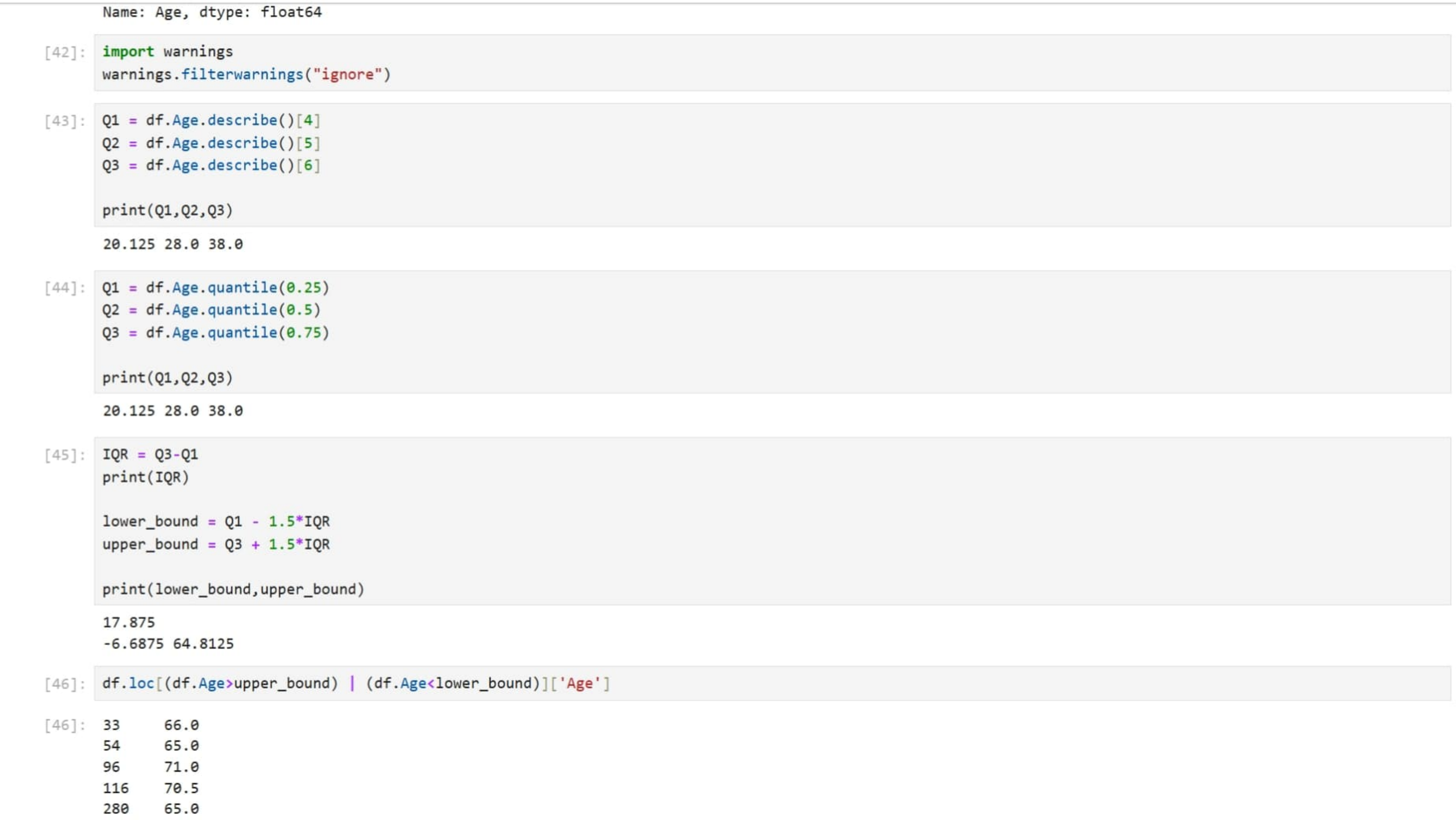
```
[24]: df.dropna(subset=['Age'],inplace=True)
```

```
[26]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 714 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  714 non-null    int64  
 1   Survived     714 non-null    int64  
 2   Pclass       714 non-null    int64  
 3   Name         714 non-null    object 
 4   Gender       714 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        714 non-null    int64  
 7   Parch        714 non-null    int64  
 8   Ticket       714 non-null    object 
 9   Fare         714 non-null    float64
 10  Cabin        185 non-null    object 
 11  Embarked     712 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 72.5+ KB
```

```
[27]: df.Age.describe()
```

```
[27]: count    714.000000
      mean      29.699118
      std       14.526497
      min        0.420000
      25%       20.125000
      50%       28.000000
      75%       38.000000
      max       80.000000
```



```
Name: Age, dtype: float64
```

[42]:
```
import warnings
warnings.filterwarnings("ignore")
```

[43]:
```
Q1 = df.Age.describe()[4]
Q2 = df.Age.describe()[5]
Q3 = df.Age.describe()[6]

print(Q1,Q2,Q3)
```

```
20.125 28.0 38.0
```

[44]:
```
Q1 = df.Age.quantile(0.25)
Q2 = df.Age.quantile(0.5)
Q3 = df.Age.quantile(0.75)

print(Q1,Q2,Q3)
```

```
20.125 28.0 38.0
```

[45]:
```
IQR = Q3-Q1
print(IQR)

lower_bound = Q1 - 1.5*IQR
upper_bound = Q3 + 1.5*IQR

print(lower_bound,upper_bound)
```

```
17.875
-6.6875 64.8125
```

[46]:
```
df.loc[(df.Age>upper_bound) | (df.Age<lower_bound)]['Age']
```

[46]:
```
33     66.0
54     65.0
96     71.0
116    70.5
280    65.0
```

[47]:

```
outlier_indices = df.loc[(df.Age>upper_bound) | (df.Age<lower_bound)].index
outlier_indices
```

[47]:

```
Index([33, 54, 96, 116, 280, 456, 493, 630, 672, 745, 851], dtype='int64')
```

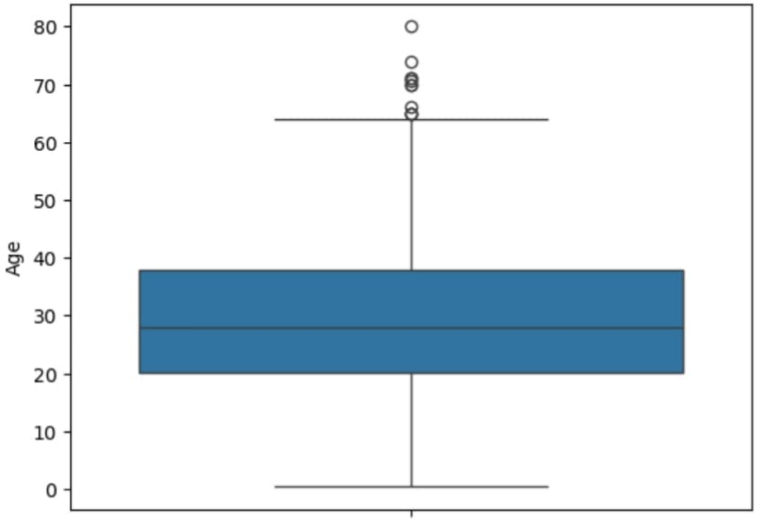
detecting outliers using boxplot

[48]:

```
sns.boxplot(df.Age)
```

[48]:

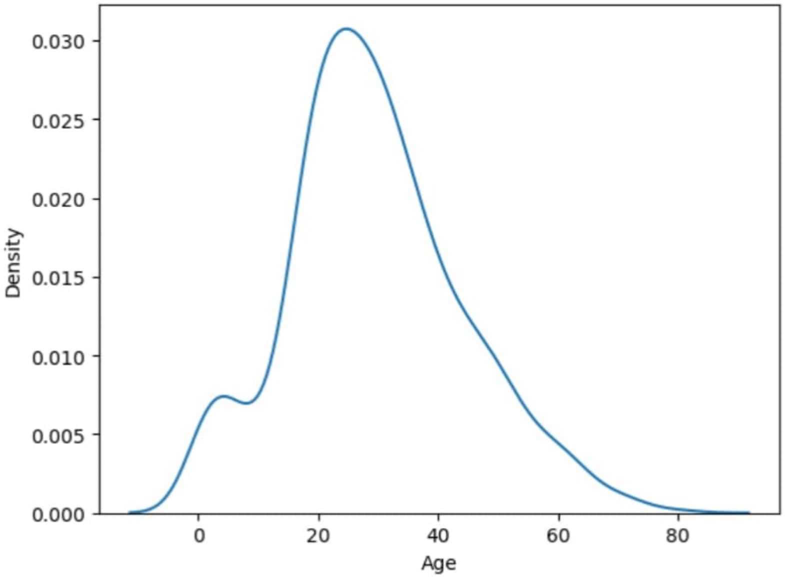
```
<Axes: ylabel='Age'>
```



[49]:

```
sns.kdeplot(df.Age)
```

```
[49]: <Axes: xlabel='Age', ylabel='Density'>
```

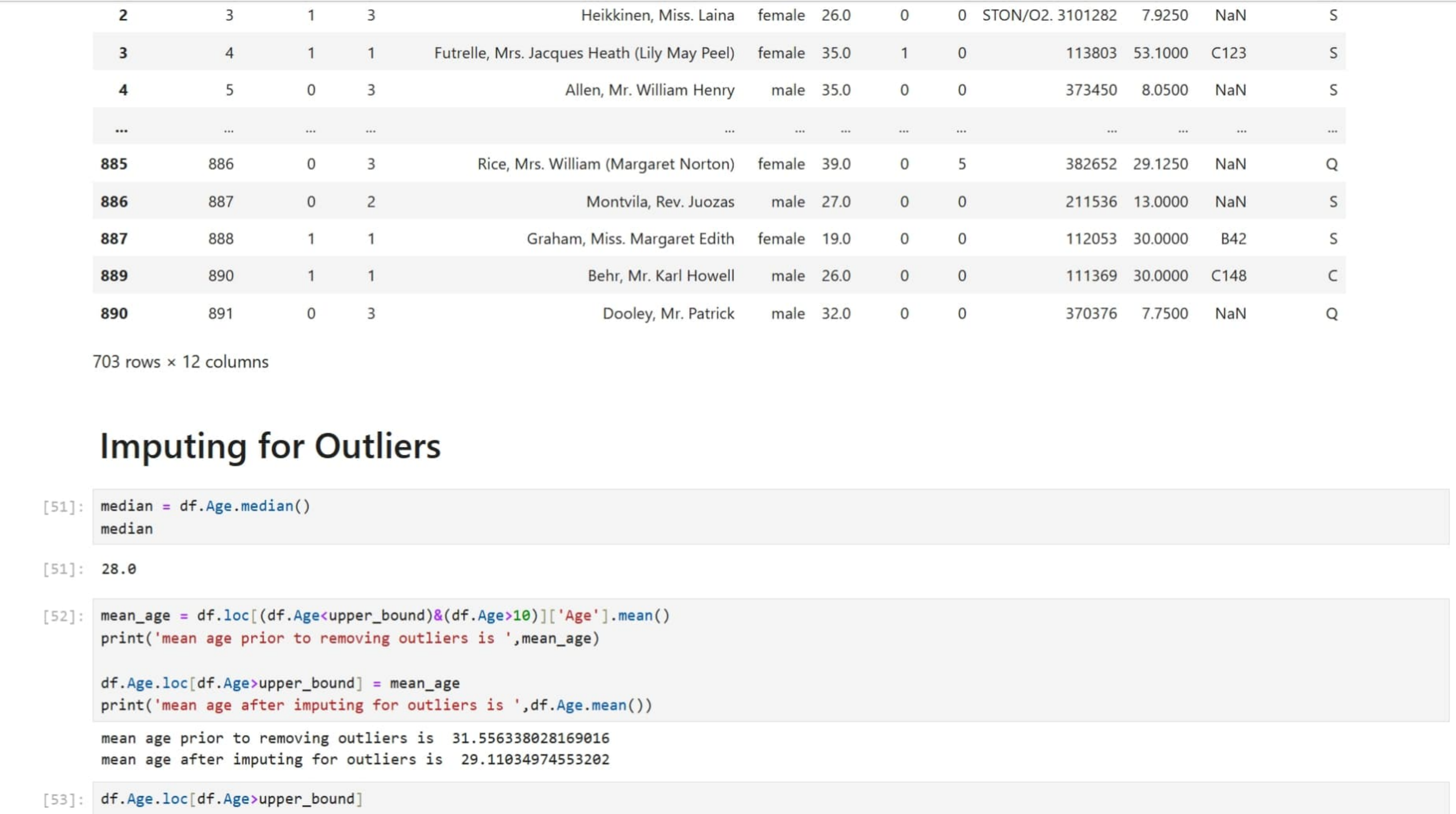


dropping the outliers

```
[50]: df.loc[~((df.Age>upper_bound) | (df.Age<lower_bound))]
```

[50]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C



2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

703 rows × 12 columns

Imputing for Outliers

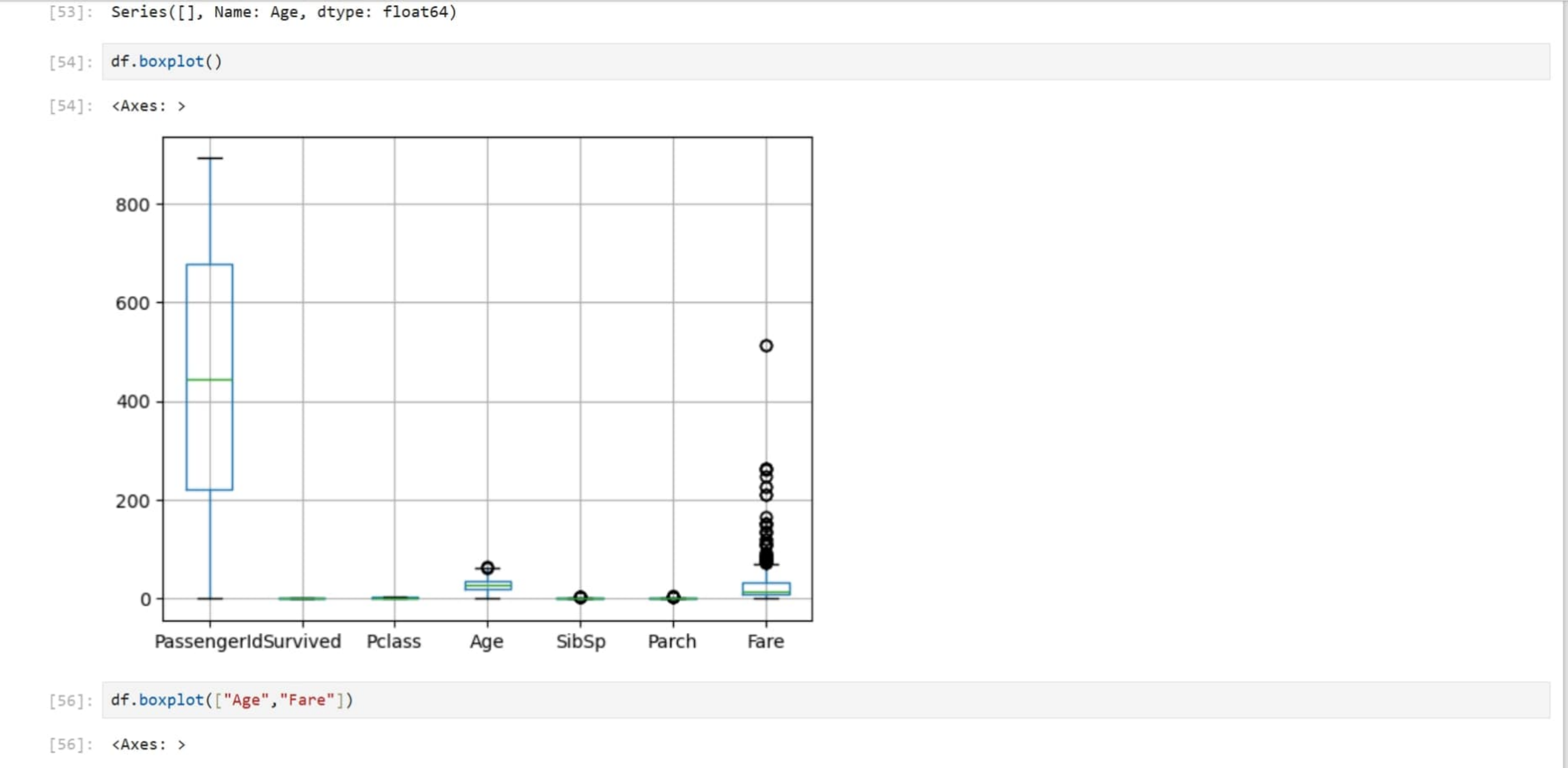
[51]:
```
median = df.Age.median()
median
```

[51]:
```
28.0
```

[52]:
```
mean_age = df.loc[(df.Age<upper_bound)&(df.Age>10)]['Age'].mean()
print('mean age prior to removing outliers is ',mean_age)

df.Age.loc[df.Age>upper_bound] = mean_age
print('mean age after imputing for outliers is ',df.Age.mean())
```

```
mean age prior to removing outliers is  31.556338028169016
mean age after imputing for outliers is  29.11034974553202
```

[53]:
```
df.Age.loc[df.Age>upper_bound]
```



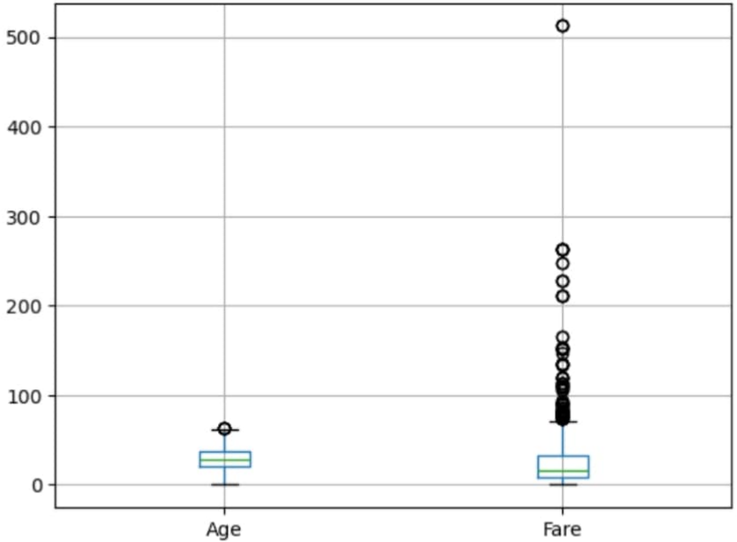
```
[53]: Series([], Name: Age, dtype: float64)
```

```
[54]: df.boxplot()
```

```
[54]: <Axes: >
```

```
[56]: df.boxplot(["Age","Fare"])
```

```
[56]: <Axes: >
```


500
400
300
200
100
0
Age
Fare

```
[ ]:
```