Handled Missing Values Part 2

July 19, 2023

0.0.1 Random Sample Imputation

Aim: Random sample imputation consists of taking random observation from the dataset and we use this observation to replace the nan values

When should it be used? It assumes that the data are missing completely at random(MCAR)

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
# load The Dataaset
```

[3]:
```
df = pd.read_csv("C:\\Users\\ssart\\Downloads\\train.csv",usecols =␣
 ↪['Age','Fare','Survived'])
```

[4]:
```
df
```

[4]:
```
     Survived   Age     Fare
0           0  22.0   7.2500
1           1  38.0  71.2833
2           1  26.0   7.9250
3           1  35.0  53.1000
4           0  35.0   8.0500
..        …     …       …
886         0  27.0  13.0000
887         1  19.0  30.0000
888         0   NaN  23.4500
889         1  26.0  30.0000
890         0  32.0   7.7500

[891 rows x 3 columns]
```

[5]:
```
df.isnull().sum()
```

[5]:
```
Survived      0
Age         177
Fare          0
```

```
dtype: int64
```

[6]:

```
#Check The Nullvalue percentage of Data
```

[7]:

```
df.isnull().mean()
```

[7]:

```
Survived    0.000000
Age         0.198653
Fare        0.000000
dtype: float64
```

[8]:

```
df['Age'].isnull().sum()
```

[8]:

```
177
```

[9]:

```
df['Age'].dropna().sample(df['Age'].isnull().sum(),random_state = 0)
```

[9]:

```
423    28.00
177    50.00
305     0.92
292    36.00
889    26.00
       …
539    22.00
267    25.00
352    15.00
99     34.00
689    15.00
Name: Age, Length: 177, dtype: float64
```

[10]:

```
df[df['Age'].isnull()].index
```

[10]:

```
Int64Index([  5,  17,  19,  26,  28,  29,  31,  32,  36,  42,
            …
            832, 837, 839, 846, 849, 859, 863, 868, 878, 888],
           dtype='int64', length=177)
```

[11]:

```
# Craete the Function For the fill missing value
```

[12]:

```
def impute_nan(df,variable,median):
    df[variable+ "_median"] = df[variable].fillna(median)
    df[variable+ "_random"] =df[variable]

    ##It will have the random sample to fill the na
    random_sample =df[variable].dropna().sample(df['Age'].isnull().
 ↪sum(),random_state=0)
    ##pandas need to have same index in order to merge the dataset
    random_sample.index=df[df[variable].isnull()].index
    df.loc[df[variable].isnull(),variable+'_random']=random_sample
```

[13]:
```
median =df.Age.median()
median
```

[13]:
```
28.0
```

[14]:
```
impute_nan(df,'Age',median)
```

[15]:
```
df.sample(10)
```

[15]:
```
     Survived   Age      Fare  Age_median  Age_random
97          1  23.0   63.3583        23.0        23.0
654         0  18.0    6.7500        18.0        18.0
714         0  52.0   13.0000        52.0        52.0
229         0   NaN   25.4667        28.0        28.0
311         1  18.0  262.3750        18.0        18.0
6           0  54.0   51.8625        54.0        54.0
670         1  40.0   39.0000        40.0        40.0
75          0  25.0    7.6500        25.0        25.0
240         0   NaN   14.4542        28.0         3.0
348         1   3.0   15.9000         3.0         3.0
```

[16]:
```
value_counts = df['Age_random'].value_counts().reset_index()
value_counts
```

[16]:
```
    index  Age_random
0    24.0          37
1    22.0          35
2    36.0          33
3    21.0          31
4    28.0          31
..    …          …
83   70.5           1
84   12.0           1
85   36.5           1
86   55.5           1
87   74.0           1

[88 rows x 2 columns]
```

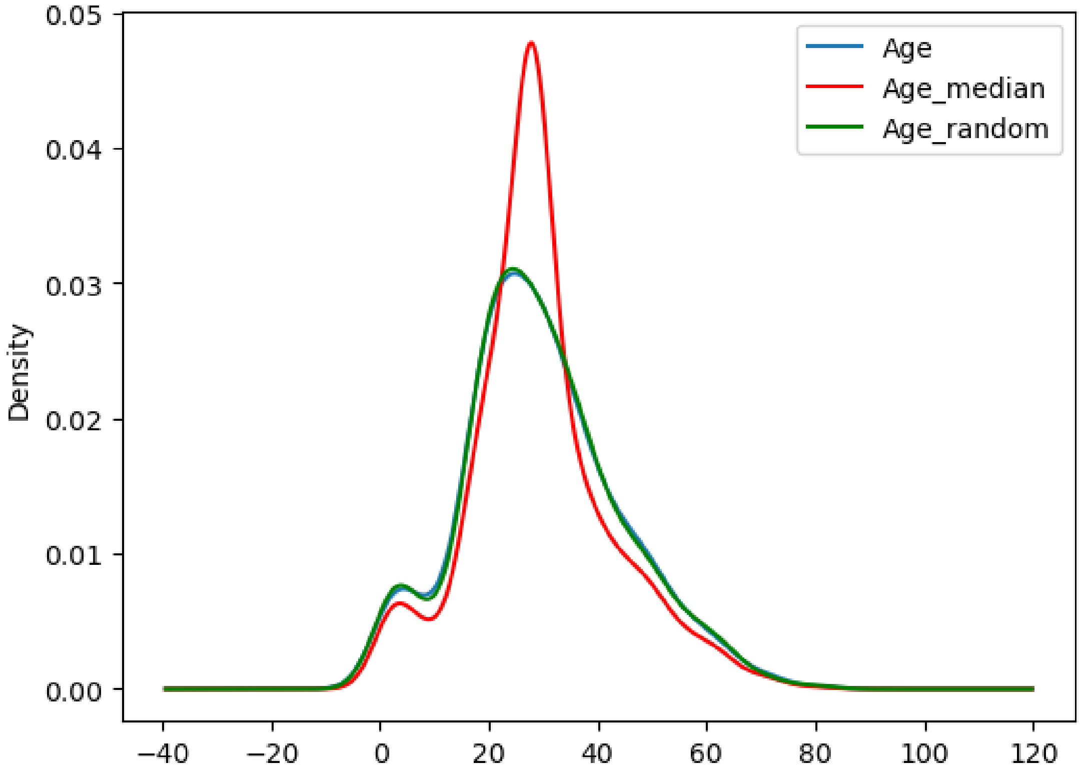
[17]:
```
# ploting the Age columns and Age_median and Age_Random
```

[18]:
```
fig=plt.figure()
fig = plt.figure()
ax = fig.add_subplot(111)
df['Age'].plot(kind='kde', ax=ax)
```

```
df.Age_median.plot(kind='kde', ax=ax, color='red')
df.Age_random.plot(kind='kde', ax=ax, color='green')
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

```
[18]: <matplotlib.legend.Legend at 0x1d2756170a0>
```

```
<Figure size 640x480 with 0 Axes>
```



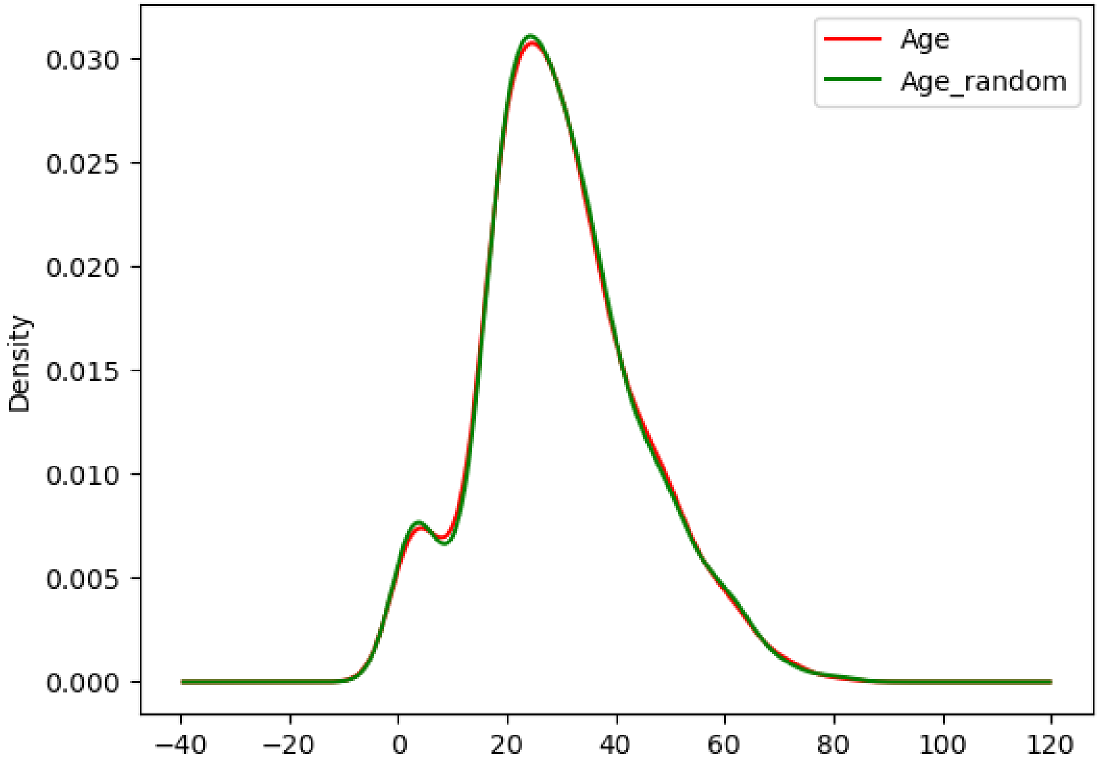
0.0.2 Advantages

1.Easy To implement 2.There is less distortion in variance ### Disadvantage Every situation randomness wont work ### Capturing NAN values with a new feature It works well if the data are not missing completely at random

```
[19]: # ploting the Age columns and Age_Random
```

```
[20]: fig = plt.figure()
      ax = fig.add_subplot(111)
      df['Age'].plot(kind='kde', ax=ax, color='red')
      df.Age_random.plot(kind='kde', ax=ax, color='green')
      lines, labels = ax.get_legend_handles_labels()
      ax.legend(lines, labels, loc='best')
```

```
[20]: <matplotlib.legend.Legend at 0x1d27678d130>
```



0.0.3 Capturing NAN values with a new feature

It works well if the data are not missing completely at random

```
[21]: df = pd.read_csv("C:\\Users\\ssart\\Downloads\\train.csv", usecols =␣
        ↪['Age','Fare','Survived'])
```

```
[22]: df
```

```
[22]:      Survived   Age     Fare
      0           0  22.0   7.2500
      1           1  38.0  71.2833
      2           1  26.0   7.9250
      3           1  35.0  53.1000
      4           0  35.0   8.0500
      ..        …     …        …
      886         0  27.0  13.0000
      887         1  19.0  30.0000
      888         0   NaN  23.4500
      889         1  26.0  30.0000
```

```
890          0  32.0   7.7500

[891 rows x 3 columns]
```

[23]:

```
df['Age_NAN'] = np.where(df['Age'].isnull(),0,1)
```

[24]:

```
df.sample(10)
```

[24]:

```
     Survived   Age      Fare  Age_NAN
839         1   NaN   29.7000        0
612         1   NaN   15.5000        0
315         1  26.0    7.8542        1
693         0  25.0    7.2250        1
341         1  24.0  263.0000        1
51          0  21.0    7.8000        1
211         1  35.0   21.0000        1
79          1  30.0   12.4750        1
886         0  27.0   13.0000        1
435         1  14.0  120.0000        1
```

[25]:

```
df.Age.median()
```

[25]:

```
28.0
```

[26]:

```
df['Age'].fillna(df.Age.median(),inplace = True)
```

[27]:

```
df.sample(10)
```

[27]:

```
     Survived   Age      Fare  Age_NAN
104         0  37.0    7.9250        1
572         1  36.0   26.3875        1
664         1  20.0    7.9250        1
164         0   1.0   39.6875        1
607         1  27.0   30.5000        1
151         1  22.0   66.6000        1
609         1  40.0  153.4625        1
419         0  10.0   24.1500        1
4           0  35.0    8.0500        1
454         0  28.0    8.0500        0
```

0.0.4 End Of Distribution

[83]:

```
df = pd.read_csv("C:\\Users\\ssart\\Downloads\\train.csv", usecols =␣
 ↪['Age','Fare','Survived'])
df.head(5)
```

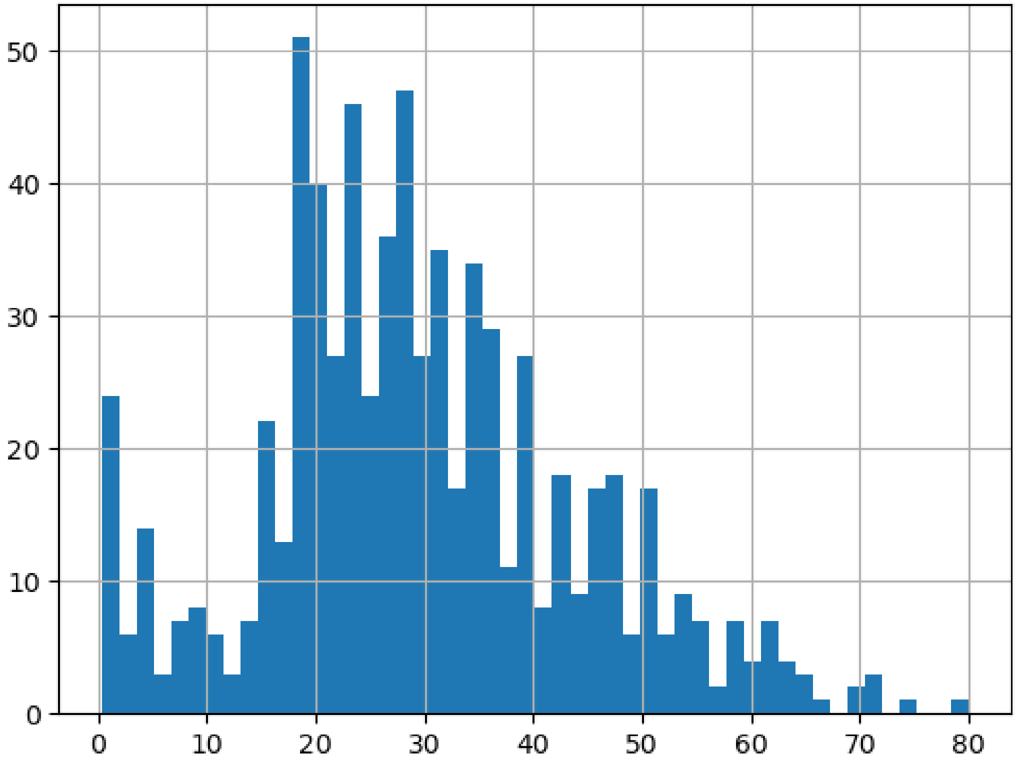
[83]:

```
   Survived   Age     Fare
0         0  22.0   7.2500
```

```
1          1  38.0  71.2833
2          1  26.0   7.9250
3          1  35.0  53.1000
4          0  35.0   8.0500
```

[88]:
```
df['Age'].hist(bins = 50)
```

[88]:
```
<AxesSubplot:>
```



[96]:
```
extreme = df.Age.mean()+3* df.Age.std()
```

[97]:
```
extreme
```

[97]:
```
73.27860964406095
```

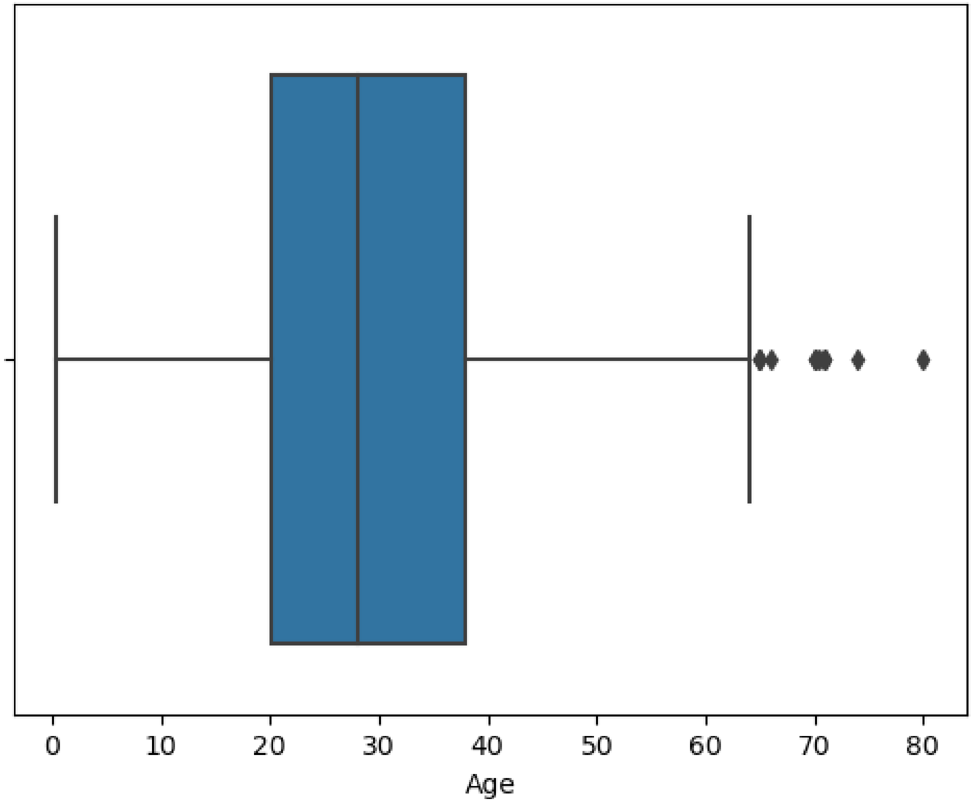
[98]:
```
import seaborn as sns
sns.boxplot('Age',data=df)
```

```
C:\Users\ssart\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
```

```
misinterpretation.
  warnings.warn(
```

[98]:

```
<AxesSubplot:xlabel='Age'>
```



[99]:

```
def impute_nan(df,variable,median,extreme):
    df[variable+"_end_distribution"]=df[variable].fillna(extreme)
    df[variable].fillna(median,inplace=True)
```

[100]:

```
impute_nan(df,'Age',df.Age.median(),extreme)
```

[106]:

```
df.sample(25)
```

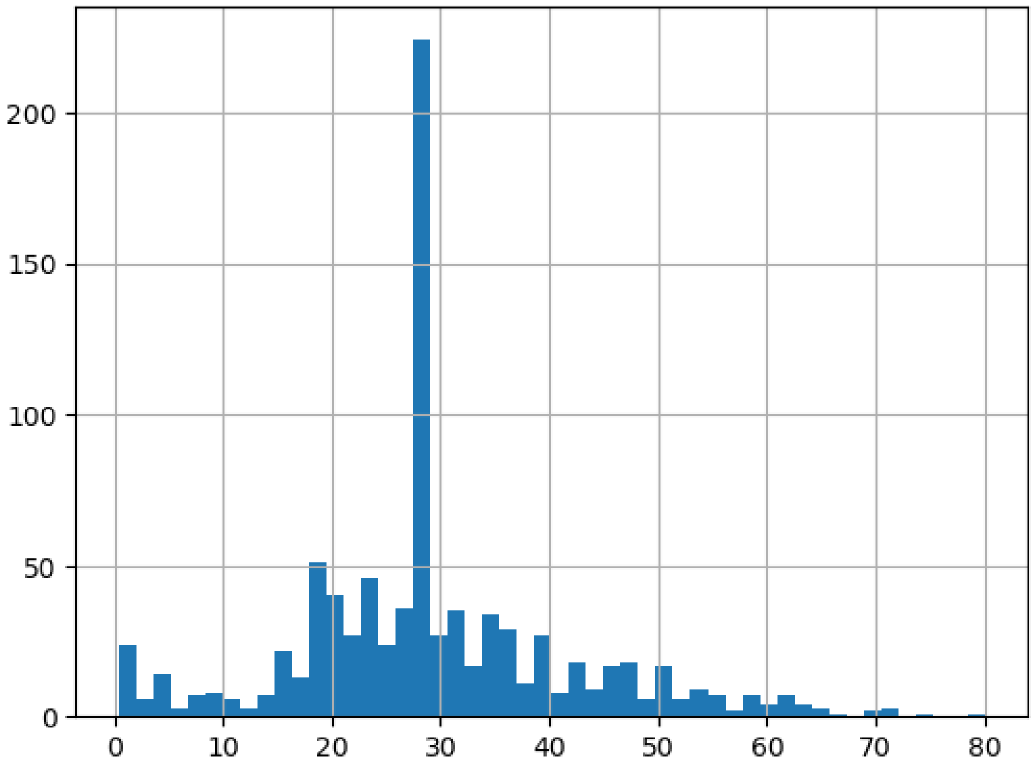
[106]:

```
     Survived   Age     Fare  Age_end_distribution
420         0  28.0   7.8958              73.27861
84          1  17.0  10.5000              17.00000
870         0  26.0   7.8958              26.00000
816         0  23.0   7.9250              23.00000
453         1  49.0  89.1042              49.00000
320         0  22.0   7.2500              22.00000
277         0  28.0   0.0000              73.27861
```

```
211         1  35.0   21.0000              35.00000
494         0  21.0    8.0500              21.00000
114         0  17.0   14.4583              17.00000
263         0  40.0    0.0000              40.00000
104         0  37.0    7.9250              37.00000
787         0   8.0   29.1250               8.00000
801         1  31.0   26.2500              31.00000
438         0  64.0  263.0000              64.00000
316         1  24.0   26.0000              24.00000
365         0  30.0    7.2500              30.00000
364         0  28.0   15.5000              73.27861
262         0  52.0   79.6500              52.00000
64          0  28.0   27.7208              73.27861
459         0  28.0    7.7500              73.27861
31          1  28.0  146.5208              73.27861
815         0  28.0    0.0000              73.27861
614         0  35.0    8.0500              35.00000
569         1  32.0    7.8542              32.00000
```

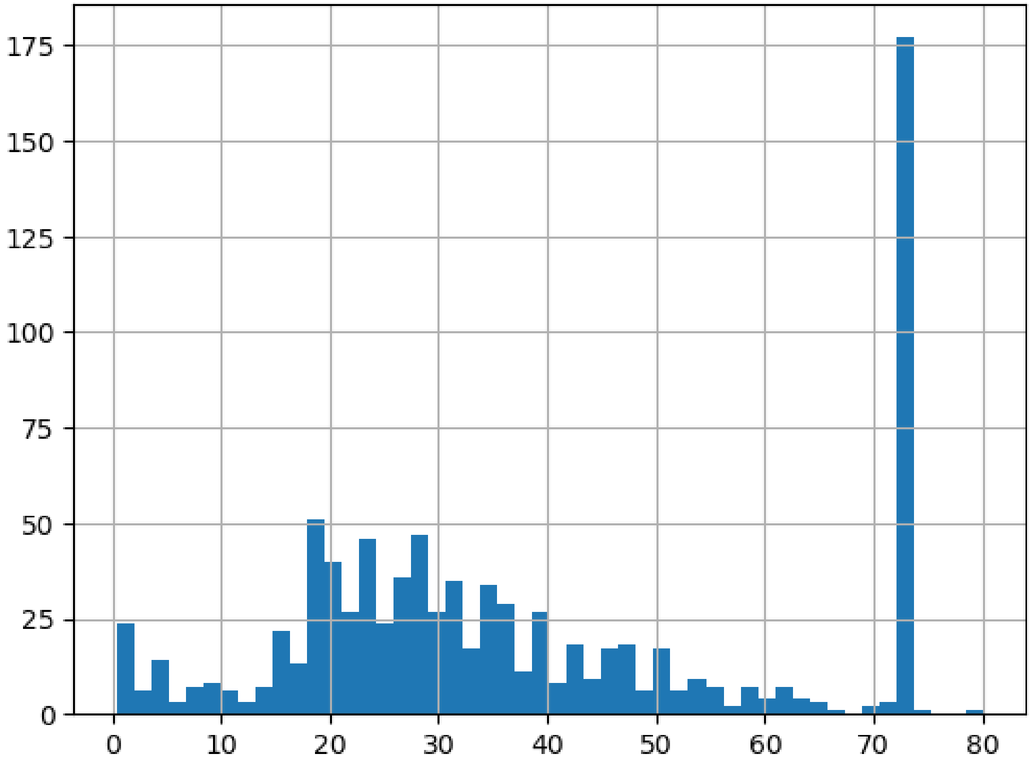
[102]:
```
df['Age'].hist(bins=50)
```

[102]:
```
<AxesSubplot:>
```



[103]:
```
df['Age_end_distribution'].hist(bins=50)
```

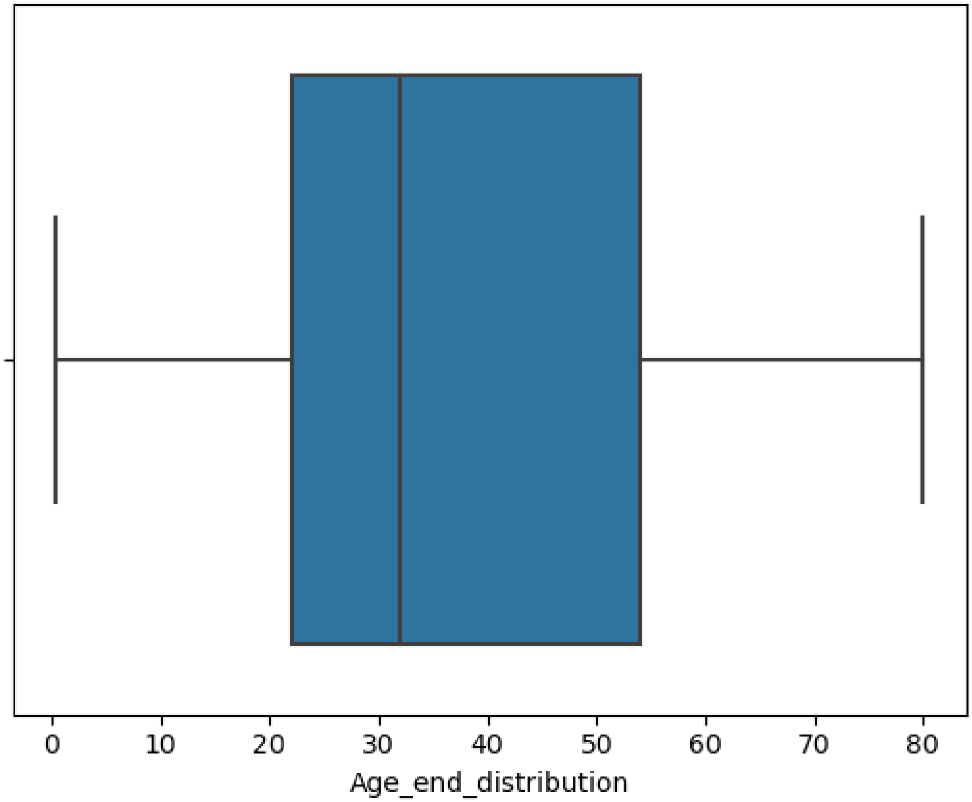
[103]:
```
<AxesSubplot:>
```



[105]:
```
sns.boxplot("Age_end_distribution",data = df)
```

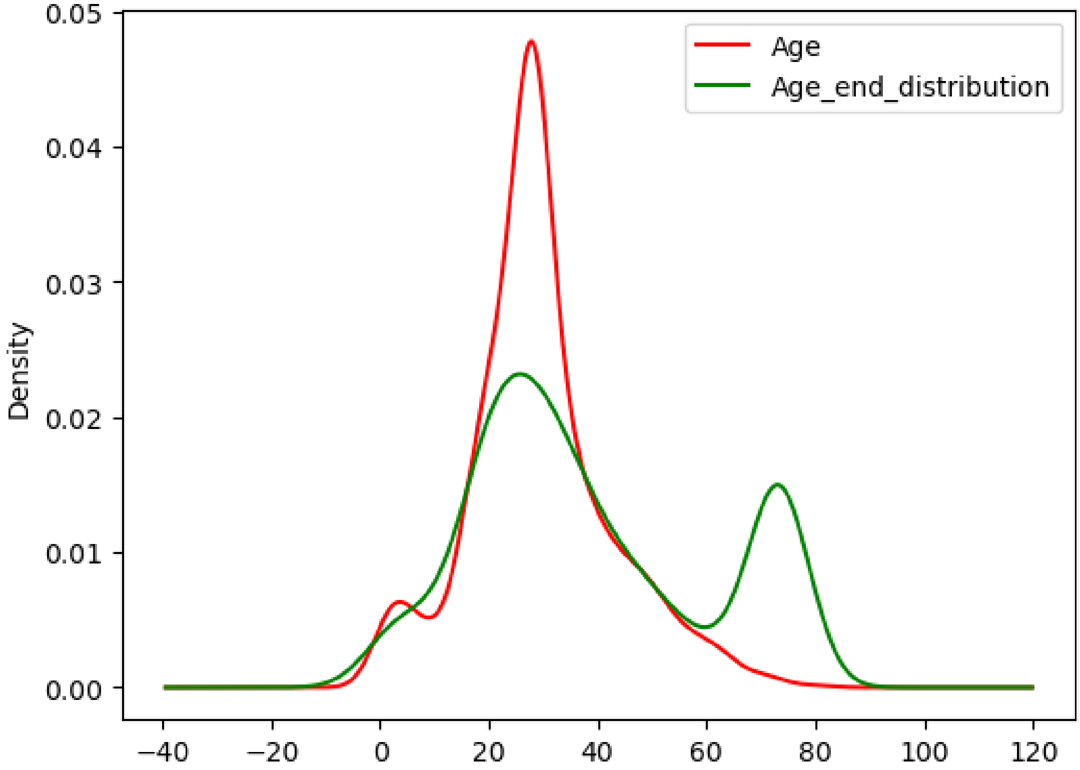
```
C:\Users\ssart\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[105]:
```
<AxesSubplot:xlabel='Age_end_distribution'>
```



```
[111]: fig = plt.figure()
       ax = fig.add_subplot(111)
       df['Age'].plot(kind='kde', ax=ax, color='red')
       df.Age_end_distribution.plot(kind='kde', ax=ax, color='green')
       lines, labels = ax.get_legend_handles_labels()
       ax.legend(lines, labels, loc='best')
```

```
[111]: <matplotlib.legend.Legend at 0x1d27aede220>
```


0.05
0.04
0.03
0.02
0.01
0.00
Density
−40
−20
0
20
40
60
80
100
120
Age
Age_end_distribution

[109]:

```
```

[ ]:

```
```