In [2]:

```
# You can import data from various sources into your Pandas
# dataframe.
# A CSV file is a type of file where each line contains a single
# record, and all the columns are separated from each other via
# a comma.
# You can read CSV files using the read_csv() function of the
# Pandas dataframe, as shown below.

import pandas as pd
titanic_data = pd.read_csv("titanic.csv")
titanic_data.head()

# If you print the dataframe header, you should see that the
# header contains first five rows
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	NaN	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	NaN	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	NaN	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	NaN	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	NaN	0	3	male	35.0	0	0	8.0500	S	Third	man	True

In [3]:

```
import pandas as pd
titanic_data = pd.read_csv("titanic.csv")
titanic_data.tail()

# If you print the dataframe tail, you should see that the
# tail contains last five rows
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	
886	NaN	0	2	male	27.0	0	0	13.00	S	Second	man	
887	NaN	1	1	female	19.0	0	0	30.00	S	First	woman	F
888	NaN	0	3	female	NaN	1	2	23.45	S	Third	woman	F
889	NaN	1	1	male	26.0	0	0	30.00	C	First	man	
890	NaN	0	3	male	32.0	0	0	7.75	Q	Third	man	

In [15]:

```
# To handle missing numerical data, we can use statistical
# techniques. The use of statistical techniques or algorithms to
# replace missing values with statistically generated values is
# called imputation.

import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams["figure.figsize"] = [8,6]
sns.set_style("darkgrid")
titanic_data = sns.load_dataset('titanic')
titanic_data.head()
```

Out[15]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

In [16]:

```
# Let’s filter some of the numeric columns from the dataset and
# see if they contain any missing values.

titanic_data = titanic_data[["survived", "pclass", "age", "fare"]]
titanic_data.head()
```

Out[16]:

	survived	pclass	age	fare
0	0	3	22.0	7.2500
1	1	1	38.0	71.2833
2	1	3	26.0	7.9250
3	1	1	35.0	53.1000
4	0	3	35.0	8.0500

In [17]:

```
# To find missing values from the aforementioned columns, you
# need to first call the isnull() method on the titanic_data
# dataframe, and then you need to call the mean() method, as
# shown below.

titanic_data.isnull().mean()

# The output shows that only the age column contains
# missing values. And the ratio of missing values is around 19.86
# percent.
```

Out[17]:

```
survived    0.000000
pclass      0.000000
age         0.198653
fare        0.000000
dtype: float64
```

In [18]:

```
# Let’s now find out the median and mean values for all the nonmissing
# values in the age column.

median = titanic_data.age.median()
print(median)
mean = titanic_data.age.mean()
print(mean)

# The age column has a median value of 28 and a mean value of
# 29.6991.
```

```
28.0
29.69911764705882
```

In [19]:

```
# To plot the kernel density plots for the actual age and median
# and mean age, we will add columns to the Pandas dataframe.

import numpy as np
titanic_data['Median_Age'] = titanic_data.age.fillna(median)
titanic_data['Mean_Age'] = titanic_data.age.fillna(mean)
titanic_data['Mean_Age'] = np.round(titanic_data['Mean_Age'], 1)
titanic_data.head(20)

# The above script adds Median_Age and Mean_Age columns
# to the titanic_data dataframe and prints the first 20 records.
# Here is the output of the above script:
```

Out[19]:

	survived	pclass	age	fare	Median_Age	Mean_Age
0	0	3	22.0	7.2500	22.0	22.0
1	1	1	38.0	71.2833	38.0	38.0
2	1	3	26.0	7.9250	26.0	26.0
3	1	1	35.0	53.1000	35.0	35.0
4	0	3	35.0	8.0500	35.0	35.0
5	0	3	NaN	8.4583	28.0	29.7
6	0	1	54.0	51.8625	54.0	54.0
7	0	3	2.0	21.0750	2.0	2.0
8	1	3	27.0	11.1333	27.0	27.0
9	1	2	14.0	30.0708	14.0	14.0
10	1	3	4.0	16.7000	4.0	4.0
11	1	1	58.0	26.5500	58.0	58.0
12	0	3	20.0	8.0500	20.0	20.0
13	0	3	39.0	31.2750	39.0	39.0
14	0	3	14.0	7.8542	14.0	14.0
15	1	2	55.0	16.0000	55.0	55.0
16	0	3	2.0	29.1250	2.0	2.0
17	1	2	NaN	13.0000	28.0	29.7
18	0	3	31.0	18.0000	31.0	31.0
19	1	3	NaN	7.2250	28.0	29.7

In [20]:

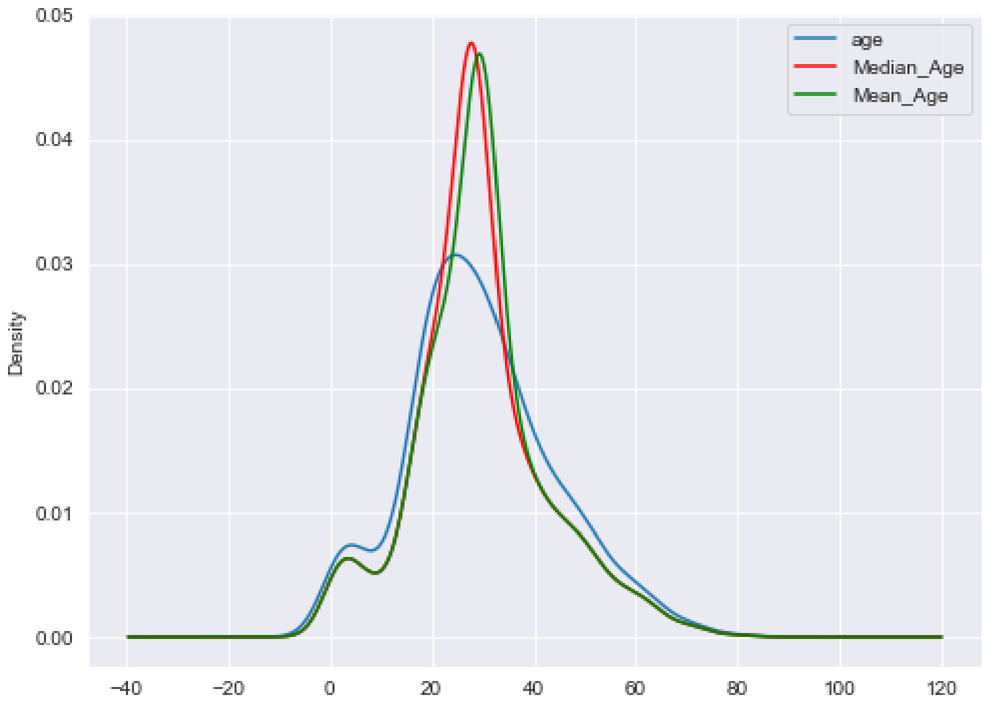
```
# Some rows in the above output show that NaN, i.e.,
# null values in the age column, have been replaced by the
# median values in the Median_Age column and by mean values
# in the Mean_Age column.
# The mean and median imputation can affect the data
# distribution for the columns containing the missing values.
# Specifically, the variance of the column is decreased by mean
# and median imputation now since more values are added to
# the center of the distribution. The following script plots the
# distribution of data for the age, Median_Age, and Mean_Age
# columns.

fig = plt.figure()
ax = fig.add_subplot(111)
titanic_data['age'] .plot(kind='kde', ax=ax)
titanic_data['Median_Age'] .plot(kind='kde', ax=ax, color='red')
titanic_data['Mean_Age'] .plot(kind='kde', ax=ax, color='green')
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')

# Here is the output of the script above:
```

Out[20]:

```
<matplotlib.legend.Legend at 0x63dfb64a30>
```



In [ ]:

```
# You can see that the default values in the age columns have
# been distorted by the mean and median imputation, and the
# overall variance of the dataset has also been decreased.

#Recommendation

# Mean and Median imputation could be used for the missing
# numerical data in case the data is missing at random. If the
# data is normally distributed, mean imputation is better, or else,
# median imputation is preferred in case of skewed
# distributions.
```