Handling Missing Data

Handling Missing Numerical Data

Mean and Median Imputation

Mean or median imputation is one of the most commonly used imputation techniques for handling missing numerical data.

```
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()
```

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

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

	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

```
titanic_data.isnull().mean()
```

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

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

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

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

```
28.0
29.69911764705882
```

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

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['Mean_Age'] = np.round(titanic_data['Mean_Age'], 1)

titanic_data.head(20)
```

```
C:\Users\usman\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\usman\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  """
C:\Users\usman\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  import sys
```

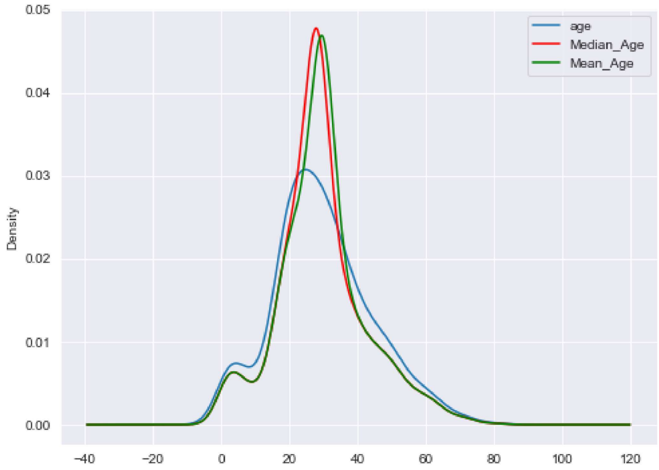
	survived	pclass	age	fare	Median_Age	Mean_Age
0	0	3	22.0	7.2500	22.0	22.0
1	1	1	38.0	71.2833	38.0	38.0
2	1	3	26.0	7.9250	26.0	26.0
3	1	1	35.0	53.1000	35.0	35.0
4	0	3	35.0	8.0500	35.0	35.0
5	0	3	NaN	8.4583	28.0	29.7
6	0	1	54.0	51.8625	54.0	54.0
7	0	3	2.0	21.0750	2.0	2.0
8	1	3	27.0	11.1333	27.0	27.0
9	1	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

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.

```
plt.rcParams["figure.figsize"] = [8,6]

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')
```

```
<matplotlib.legend.Legend at 0x2181c88f948>
```



You can clearly 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.

Mean and Median imputation could be used for 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.

Disadvantages

As said earlier, the biggest disadvantage of mean and median imputation is that it affects the default data distribution and variance and covariance of the data.

End of Distribution Imputation

The mean and median imputation are not good techniques for missing value imputations in case the data is not randomly missing.

For randomly missing data, the most commonly used techniques are end of distribution/ end of tail imputation. In the end of tail imputation, a value is chosen from the tail end of the data. This value signifies that the actual data for the record was missing.

Hence, data that is not randomly missing can be taken to account while training statistical models on the data.

```
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')
```

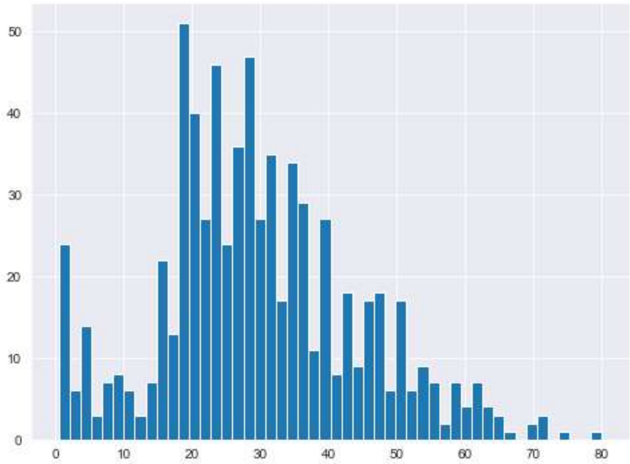
```
titanic_data  = titanic_data[["survived", "pclass", "age", "fare"]]
```

```
titanic_data.isnull().mean()
```

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

```
titanic_data.age.hist(bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2181c9c17c8>
```



The output shows that the age column has an almost normal distribution. Hence, the end of the distribution value can be calculated by multiplying the mean value of the age column by three standard deviations.

```
eod_value = titanic_data.age.mean() + 3 * titanic_data.age.std()
print(eod_value)
```

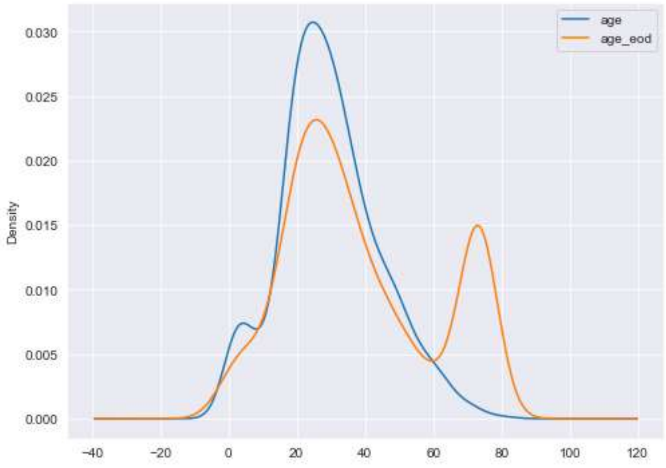
```
73.27860964406095
```

```
import numpy as np

titanic_data['age_eod'] = titanic_data.age.fillna(eod_value)
titanic_data.head(20)
```

	survived	pclass	age	fare	age_eod
0	0	3	22.0	7.2500	22.00000
1	1	1	38.0	71.2833	38.00000
2	1	3	26.0	7.9250	26.00000
3	1	1	35.0	53.1000	35.00000
4	0	3	35.0	8.0500	35.00000
5	0	3	NaN	8.4583	73.27861
6	0	1	54.0	51.8625	54.00000
7	0	3	2.0	21.0750	2.00000
8	1	3	27.0	11.1333	27.00000
9	1	2	14.0	30.0708	14.00000
10	1	3	4.0	16.7000	4.00000
11	1	1	58.0	26.5500	58.00000
12	0	3	20.0	8.0500	20.00000
13	0	3	39.0	31.2750	39.00000
14	0	3	14.0	7.8542	14.00000
15	1	2	55.0	16.0000	55.00000
16	0	3	2.0	29.1250	2.00000
17	1	2	NaN	13.0000	73.27861
18	0	3	31.0	18.0000	31.00000
19	1	3	NaN	7.2250	73.27861

```
plt.rcParams["figure.figsize"] = [8,6]

fig = plt.figure()
ax = fig.add_subplot(111)

titanic_data['age'] .plot(kind='kde', ax=ax)
titanic_data['age_eod'] .plot(kind='kde', ax=ax)
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

```
<matplotlib.legend.Legend at 0x2181cb0be88>
```



One of the main advantages of the end of distribution imputation is that it can be applied to the dataset where values are not missing at random.

The other advantages of end of distribution imputation include its simplicity to understand, ability to create big datasets in a short time, and applicability in the production environment.

Aribitrary Value 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')
```

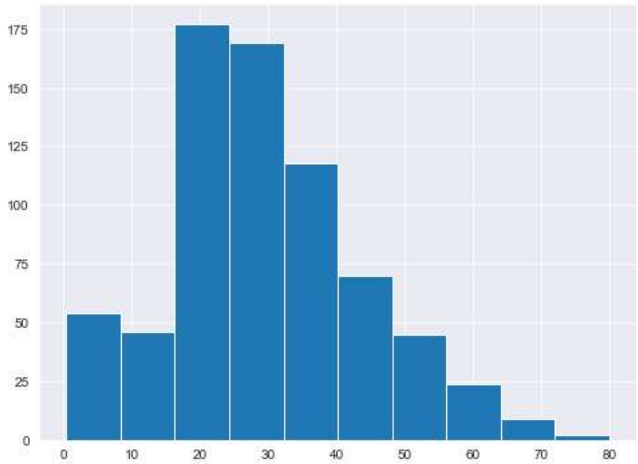
```
titanic_data  = titanic_data[["survived", "pclass", "age", "fare"]]
```

```
titanic_data.isnull().mean()
```

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

```
titanic_data.age.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2181c883e88>
```




```
import numpy as np
```

```
titanic_data['age_99'] = titanic_data.age.fillna(99)
```

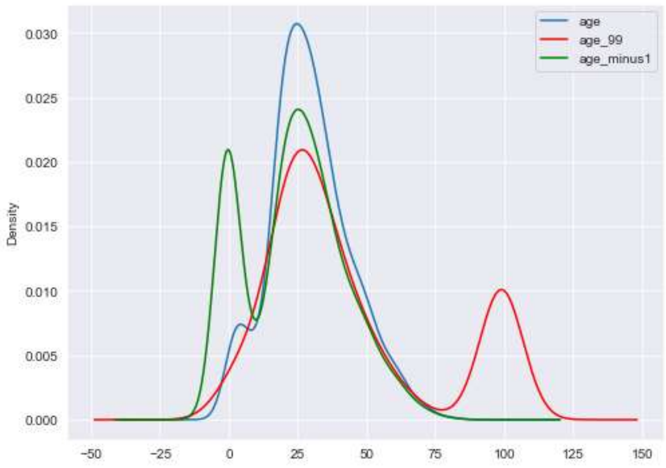
```
titanic_data['age_minus1'] = titanic_data.age.fillna(-1)
```

```
titanic_data.head(20)
```

	survived	pclass	age	fare	age_99	age_minus1
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	99.0	-1.0
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	99.0	-1.0
18	0	3	31.0	18.0000	31.0	31.0
19	1	3	NaN	7.2250	99.0	-1.0

```
plt.rcParams["figure.figsize"] = [8,6]

fig = plt.figure()
ax = fig.add_subplot(111)

titanic_data['age'] .plot(kind='kde', ax=ax)

titanic_data['age_99'] .plot(kind='kde', ax=ax, color='red')

titanic_data['age_minus1'] .plot(kind='kde', ax=ax, color='green')

lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

```
<matplotlib.legend.Legend at 0x2181c9dfb08>
```



Handling Categorical Data

Frequent Category Imputation