Practical2A

Aim:- Demonstrate data imputation with statistical technique on numerical values and write down the conclusion about the assumption

In [77]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
df = pd.read_csv("titanic_toy - titanic_toy.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Age	Fare	Family	Survived
0	22.0	7.2500	1	0
1	38.0	71.2833	1	1
2	26.0	7.9250	0	1
3	35.0	53.1000	1	1
4	35.0	8.0500	0	0

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Age       714 non-null    float64
 1   Fare      846 non-null    float64
 2   Family    891 non-null    int64  
 3   Survived  891 non-null    int64  
dtypes: float64(2), int64(2)
memory usage: 28.0 KB
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Age         177
Fare         45
Family        0
Survived      0
dtype: int64
```

In [10]:
```
df.isnull().mean()*100
```

Out[10]:
```
Age         19.865320
Fare         5.050505
Family       0.000000
Survived     0.000000
dtype: float64
```

In [11]:
```
x = df.drop(columns = "Survived") # Independent columns
```

In [12]:
```
x
```

Out[12]:

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

891 rows × 3 columns

In [13]:
```
y = df['Survived']
```

In [14]:
```
y # dependent columns
```

Out[14]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [15]:
```
df.shape
```

Out[15]:
```
(891, 4)
```

In [16]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
X_train,X_test,Y_train,Y_test = train_test_split(x,y,test_size=0.2,random_state=2)
```

In [21]:
```
X_train.shape
```

Out[21]:
```
(712, 3)
```

In [22]:
```
X_test.shape
```

Out[22]:
```
(179, 3)
```

In [24]:
```
df.describe()
```

Out[24]:

	Age	Fare	Family	Survived
count	714.000000	846.000000	891.000000	891.000000
mean	29.699118	32.279338	0.904602	0.383838
std	14.526497	50.305796	1.613459	0.486592
min	0.420000	0.000000	0.000000	0.000000
25%	20.125000	7.895800	0.000000	0.000000
50%	28.000000	14.454200	0.000000	0.000000
75%	38.000000	31.206250	1.000000	1.000000
max	80.000000	512.329200	10.000000	1.000000

In [57]:
```
mean_age_train = X_train['Age'].mean()
median_age_train = X_train['Age'].median()
var_age_train = X_train['Age'].var()
```

In [58]:
```
mean_age_train
```

Out[58]: 29.78590425531915

In [59]:
```
median_age_train
```

Out[59]: 28.75

In [60]:
```
var_age_train
```

Out[60]: 204.3495133904614

In [89]:
```
mean_fare_train = X_train['Fare'].mean()
median_fare_train = X_train['Fare'].median()
var_fare_train = X_train['Fare'].var()
```

In [90]:
```
mean_fare_train
```

Out[90]: 32.617596893491076

In [91]:
```
median_fare_train
```

Out[91]: 14.4583

In [92]:
```
var_fare_train
```

Out[92]: 2448.197913706318

In [115]:
```
mean_age_test = X_test['Age'].mean()
median_age_test = X_test['Age'].median()
var_age_test = X_test['Age'].var()
```

In [116]:
```
mean_age_test
```

Out[116]: 29.3728

In [117]:
```
median_age_test
```

Out[117]: 28.0

In [68]:
```
var_age_test
```

Out[68]: 2872.7824991474345

In [69]:
```
mean_fare_test = X_test['Fare'].mean()
median_fare_test = X_test['Fare'].median()
var_fare_test = X_test['Fare'].var()
```

In [70]:
```
mean_fare_test
```

Out[70]: 30.934262941176478

In [71]:
```
median_fare_test
```

Out[71]: 13.89585

In [72]:
```
var_fare_test
```

Out[72]: 2872.7824991474345

In [78]:
```
X_train['Age_mean'] = X_train['Age'].fillna(mean_age_train)
X_train['Age_median'] = X_train['Age'].fillna(median_age_train)
```

In [94]:
```
X_train['Fare_mean'] = X_train['Fare'].fillna(mean_fare_train)
X_train['Fare_median'] = X_train['Fare'].fillna(median_fare_train)
```

In [95]:
```
X_train
```

Out[95]:

	Age	Fare	Family	Age_mean	Age_median	Fare_mean	Fare_median
30	40.0	27.7208	0	40.000000	40.00	27.7208	27.7208
10	4.0	16.7000	2	4.000000	4.00	16.7000	16.7000
873	47.0	9.0000	0	47.000000	47.00	9.0000	9.0000
182	9.0	31.3875	6	9.000000	9.00	31.3875	31.3875
876	20.0	9.8458	0	20.000000	20.00	9.8458	9.8458
...	...	...	...	...	...	...	...
534	30.0	8.6625	0	30.000000	30.00	8.6625	8.6625
584	NaN	8.7125	0	29.785904	28.75	8.7125	8.7125
493	71.0	49.5042	0	71.000000	71.00	49.5042	49.5042
527	NaN	221.7792	0	29.785904	28.75	221.7792	221.7792
168	NaN	25.9250	0	29.785904	28.75	25.9250	25.9250

712 rows × 7 columns

In [82]:
```
print("Before imputation variance of age",X_train['Age'].var())
print("After imputation varaince of mean age",X_train['Age_mean'].var())
print("After imputation varaince of median age",X_train['Age_median'].var())
```

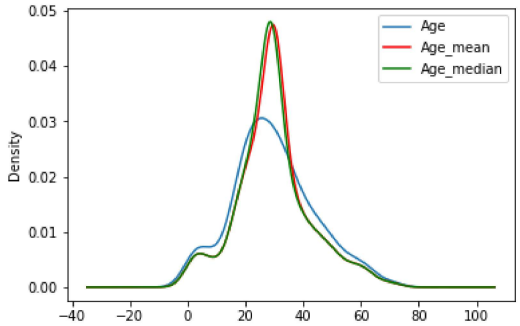
```
Before imputation variance of age 204.3495133904614
After imputation varaince of mean age 161.81262452718673
After imputation varaince of median age 161.9895663346054
```

In [96]:
```
print("Before imputation variance of fare",X_train['Fare'].var())
print("After imputation varaince of mean fare",X_train['Fare_mean'].var())
print("After imputation varaince of median fare",X_train['Fare_median'].var())
```

```
Before imputation variance of fare 2448.197913706318
After imputation varaince of mean fare 2324.2385256705547
After imputation varaince of median fare 2340.0910219753637
```

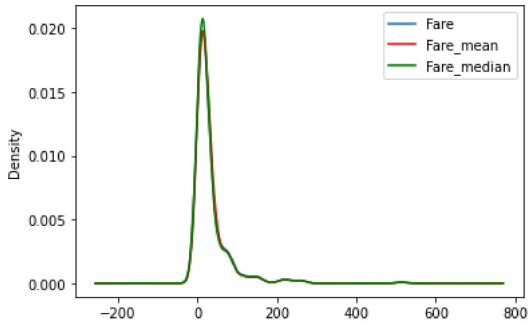
In [88]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111)
X_train['Age'].plot(kind='kde',ax=ax)
X_train['Age_mean'].plot(kind='kde',ax=ax,color='red')
X_train['Age_median'].plot(kind='kde',ax=ax,color='green')
lines,labels = ax.get_legend_handles_labels()
ax.legend(lines,labels,loc='best')
```

Out[88]:
```
<matplotlib.legend.Legend at 0x248b0f98fa0>
```



In [97]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111)
X_train['Fare'].plot(kind='kde',ax=ax)
X_train['Fare_mean'].plot(kind='kde',ax=ax,color='red')
X_train['Fare_median'].plot(kind='kde',ax=ax,color='green')
lines,labels = ax.get_legend_handles_labels()
ax.legend(lines,labels,loc='best')
```

Out[97]: <matplotlib.legend.Legend at 0x248b10d9a90>



In [100]:
```
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

In [101]:
```
imputer1 = SimpleImputer(strategy='mean')
imputer2 = SimpleImputer(strategy='median')
```

In [107]:
```
trf = ColumnTransformer([
('imputer1',imputer1,['Age']),
('imputer2',imputer2,['Fare'])
],remainder='passthrough')
```

In [108]:
```
trf.fit(df)
```

Out[108]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('imputer1', SimpleImputer(), ['Age']),
                                ('imputer2', SimpleImputer(strategy='median'),
                                 ['Fare'])])
```

In [109]:
```
trf.named_transformers_['imputer1'].statistics_
```

Out[109]:
```
array([29.69911765])
```

In [110]:
```
trf.named_transformers_['imputer2'].statistics_
```

Out[110]:
```
array([14.4542])
```

In [111]:
```
df
```

Out[111]:

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

891 rows × 4 columns

In [113]:
```
sm =  trf.transform(df)
```

In [114]:
```
sm
```

Out[114]:
```
array([[22.        ,  7.25      ,  1.        ,  0.        ],
       [38.        , 71.2833    ,  1.        ,  1.        ],
       [26.        ,  7.925     ,  0.        ,  1.        ],
       ...,
       [29.69911765, 23.45      ,  3.        ,  0.        ],
       [26.        , 14.4542    ,  0.        ,  1.        ],
       [32.        ,  7.75      ,  0.        ,  0.        ]])
```

Conclusion:- As per the observation age and fare Columns have null values more than 5%, we can fill those values through statistical technique using mean median

In [ ]:
```

```