Aim : Demonstrate data imputation with Statistical techniques on numerical value and write down the conclusion about the assumption.

In [1]:
```
import pandas as pd
import numpy as np
import os
```

In [2]:
```
df=pd.read_csv("titanic_toy.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	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 [4]:
```
df.tail()
```

Out[4]:

	Age	Fare	Family	Survived
886	27.0	13.00	0	0
887	19.0	30.00	0	1
888	NaN	23.45	3	0
889	26.0	NaN	0	1
890	32.0	7.75	0	0

In [5]:
```
print(df)
```

```
      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 x 4 columns]
```

In [7]:
```
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 [9]:
```
df.isnull().sum()
```

Out[9]:
```
Age         177
Fare         45
Family        0
Survived      0
dtype: int64
```

In [11]:
```
df.isnull().mean()*100
```

Out[11]:
```
Age         19.865320
Fare         5.050505
Family       0.000000
Survived     0.000000
dtype: float64
```

In [14]:
```
x=df.drop(columns=["Survived"]) # indepentent columns
df
```

Out[14]:

	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 [16]:
```
y=df["Survived"] # dependent columns
df
```

Out[16]:

	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 [17]:
```
df.size
```

Out[17]:
```
3564
```

In [19]:
```
df.shape
```

Out[19]:
```
(891, 4)
```

In [20]:
```
from sklearn.model_selection import train_test_split
```

In [24]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=11)
```

In [25]:
```
x_train.shape
```

Out[25]:
```
(712, 3)
```

In [26]:
```
x_test.shape
```

Out[26]:
```
(179, 3)
```

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

Out[27]:

	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 [48]:
```
# Age ka mean and median
mean_age = x_train["Age"].mean()
median_age = x_train["Age"].median()
```

In [104]:
```
mean_age
```

Out[104]: 29.605830449826986

In [50]:
```
# Fare ka mean and median
mean_fare = x_train["Fare"].mean()
median_fare = x_train["Fare"].median()
```

In [105]:
```
mean_fare
```

Out[105]: 33.15548921713435

In [109]:
```
mean_family = x_train["Family"].mean()
mean_famaily= x_train["Family"].median()
```

In [110]:
```
mean_family
```

Out[110]: 0.898876404494382

In [106]:
```
x_train ["Age_mean"]=x_train["Age"].fillna(mean_age)
x_train["Age_median"]=x_train["Age"].fillna(median_age)
```

In [107]:
```
x_train ["Fare_mean"]=x_train["Fare"].fillna(mean_fare)
x_train["Fare_median"]=x_train["Fare"].fillna(median_fare)
```

In [108]:
```
x_train
```

Out[108]:

	Age	Fare	Family	Age_mean	Age_median	Fare_mean	Fare_median
333	16.0	18.0000	2	16.0	16.0	18.000000	18.0000
662	47.0	NaN	0	47.0	47.0	33.155489	14.4583
382	32.0	7.9250	0	32.0	32.0	7.925000	7.9250
331	45.5	28.5000	0	45.5	45.5	28.500000	28.5000
149	42.0	13.0000	0	42.0	42.0	13.000000	13.0000
...	...	...	...	...	...	...	...
269	35.0	135.6333	0	35.0	35.0	135.633300	135.6333
337	41.0	134.5000	0	41.0	41.0	134.500000	134.5000
91	20.0	7.8542	0	20.0	20.0	7.854200	7.8542
80	22.0	9.0000	0	22.0	22.0	9.000000	9.0000
703	25.0	7.7417	0	25.0	25.0	7.741700	7.7417

712 rows × 7 columns

In [58]:
```
print("Before impution variance of age",x_train["Age"].var())
print("After imputation variance of mean age",x_train["Age_mean"].var())
print("After imputation variance of median",x_train["Age_median"].var())
```

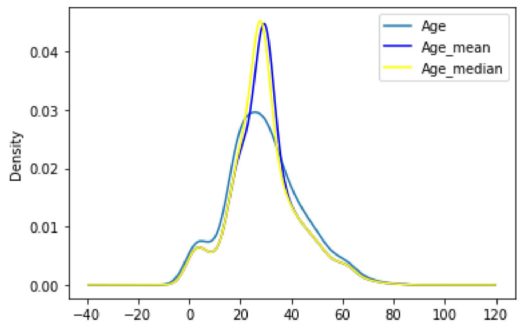
```
Before impution variance of age 213.51728050499824
After imputation variance of mean age 173.27633031136986
After imputation variance of median 173.67086248024583
```

In [59]:
```
print("Before impution variance of fare",x_train["Fare"].var())
print("After imputation variance of mean fare",x_train["Fare_mean"].var())
print("After imputation variance of median",x_train["Fare_median"].var())
```

```
Before impution variance of fare 2686.9632753477113
After imputation variance of mean fare 2554.6936345078097
After imputation variance of median 2571.0565152445192
```

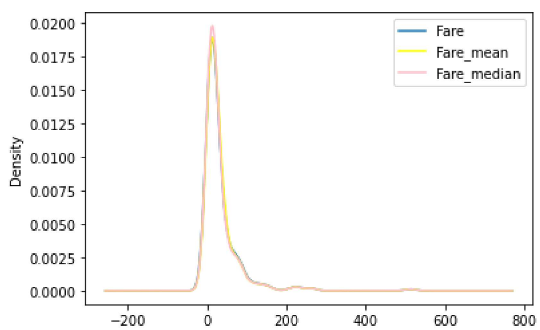
In [71]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax=fig.add_subplot(111)
x_train["Age"].plot(kind="kde",ax= ax) # original distribution
# After 3imputation with mean
x_train["Age_mean"].plot(kind="kde",ax=ax,color="blue")
#After imputation with median
x_train["Age_median"].plot(kind="kde",ax=ax,color="yellow")
# adding Legends
lines,labels= ax.get_legend_handles_labels()
ax.legend(lines,labels,loc="best")
```

Out[71]: <matplotlib.legend.Legend at 0x14ff919e250>



In [103]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax=fig.add_subplot(111)
x_train["Fare"].plot(kind="kde",ax= ax) # original distribution
# After 3imputation with mean
x_train["Fare_mean"].plot(kind="kde",ax=ax,color="yellow")
#After imputation with median
x_train["Fare_median"].plot(kind="kde",ax=ax,color="pink")
# adding Legends
lines,labels= ax.get_legend_handles_labels()
ax.legend(lines,labels,loc="best")
```

Out[103]: <matplotlib.legend.Legend at 0x14ffc4ab2e0>



In [83]:
```
import numpy as np

# Importing the SimpleImputer class
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

In [89]:
```
imputer1 =SimpleImputer(strategy="mean")
imputer2 =SimpleImputer(strategy="median")
```

In [90]:
```
trf = ColumnTransformer([
    ("imputer1",imputer1,["Age"]),
    ("imputer2",imputer2,["Fare"]),

],remainder="passthrough")
```

In [92]:
```
trf.fit(df)
```

Out[92]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('imputer1', SimpleImputer(), ['Age']),
                                ('imputer2', SimpleImputer(strategy='median'),
                                 ['Fare'])])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [94]:
```
trf.named_transformers_["imputer1"].statistics_
```

Out[94]:
```
array([29.69911765])
```

In [96]:
```
sm = trf.transform(df)
```

In [97]:
```
sm
```

Out[97]:
```
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.        ]])
```

if data is consists more than 5% then then there is lots of Variance

In [ ]: