Advanzed Programming assignment4 WanwenLiu & Sunhe

In [1]:
```
import numpy as np
#create X and y randomly
y= np.random.uniform(0,100,6)
y
```

Out[1]:
```
array([91.95607737,  1.51124101, 35.81806915, 16.7697369 , 70.51854637,
       86.76181452])
```

In [2]:
```
a=np.random.uniform(0,100,36)
X=a.reshape(6,6)
X
```

Out[2]:
```
array([[57.11513683, 58.10198887, 85.21629453, 73.14228742, 51.83102778,
        70.86194672],
       [58.33458246, 75.6627765 , 88.039993  , 72.7732583 , 64.36802645,
        48.98773688],
       [85.23376156, 81.21813896, 84.39137076, 18.43080569, 99.05140136,
         5.02718552],
       [50.85677313,  7.72544052, 18.2755791 , 96.31172433, 40.91368942,
        46.37178665],
       [34.1643108 , 96.13128125, 15.11786897, 59.14814663,  7.13787875,
        87.68900349],
       [36.77072695,  7.07948098, 83.76354929, 16.22301834, 87.14510033,
        42.62549492]])
```

In [3]:
```
selectcolumns=np.random.choice(6, 6,replace=False)
selectcolumns
```

Out[3]:
```
array([1, 3, 2, 4, 0, 5])
```

In [6]:
```
from numpy import NaN
import random
#randomly set nan values
for column in selectcolumns:
    row=random.randint(0, 5)
    X[row,column]=np.nan
```

In [7]:
```
X
```

Out[7]:
```
array([[       nan,        nan, 85.21629453, 73.14228742, 51.83102778,
        70.86194672],
       [58.33458246, 75.6627765 , 88.039993  ,        nan, 64.36802645,
        48.98773688],
       [85.23376156, 81.21813896, 84.39137076,        nan, 99.05140136,
         5.02718552],
       [50.85677313,  7.72544052, 18.2755791 , 96.31172433, 40.91368942,
        46.37178665],
       [34.1643108 ,        nan,        nan, 59.14814663,        nan,
        87.68900349],
       [36.77072695,  7.07948098, 83.76354929, 16.22301834, 87.14510033,
               nan]])
```

In [8]:
```
#SimpleImputerQuartile
from sklearn.base import TransformerMixin
class SimpleImputerQuartile(TransformerMixin):
    def _init_(self):
        pass

    def fit(self, X, y=None):
        self.sigma = np.nanvar(X, axis = 0)
        self.mu = np.nanmean(X, axis=0)
        return(self)

    def transform(self, X):
        for i in range(X.shape[0]):
            for j in range(X.shape[1]):
                if(np.isnan(X[i,j])):
                    X[i,j]= np.random.normal(loc=self.mu[j],scale=self.sigma[j],size=None)
        return(X)

quartile= SimpleImputerQuartile()
quartile= quartile.fit(X,y)
XX=quartile.transform(X)
print(quartile.mu)
print(quartile.sigma)
print(XX)
```

```
[53.07203098 42.92145924 71.93735734 61.20629418 68.66184907 51.78753185]
[ 338.04262495 1265.50916819  722.03455169  850.64755928  467.36585751
  775.28962116]
[[  -62.1798176    521.6641359     85.21629453    73.14228742
    51.83102778    70.86194672]
 [   58.33458246    75.6627765     88.039993    -282.84690303
    64.36802645    48.98773688]
 [   85.23376156    81.21813896    84.39137076   852.73972034
    99.05140136     5.02718552]
 [   50.85677313     7.72544052    18.2755791     96.31172433
    40.91368942    46.37178665]
 [   34.1643108  -2079.19050903  1358.67862551    59.14814663
    60.59492164    87.68900349]
 [   36.77072695     7.07948098    83.76354929    16.22301834
    87.14510033   139.65041592]]
```

In [9]:
```
#pipeline with knn
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor()
knnpipeline = Pipeline([('quartile', quartile),('knn', knn)])
pipeline = knnpipeline.fit(X,y)
yknnpredict= knnpipeline.predict(X)
print(yknnpredict)
```

```
[46.56338779 46.56338779 46.56338779 46.56338779 42.27588159 46.56338779]
```

In [ ]: