In [ ]:
```
#Roll No: 22150
#NAME: Neha Kamalakar Nemade
#CLASS: SE-5(G-5)
# SUBJECT:Data analytics
#Data wrangling
```

In [ ]:
```
# Pre-processing Data in Python
# Dealing with Missing Values in Python
# Data Formatting in Python
# Data Normalization in Python
# Binning in Python
# Turning categorical variables into quantitative variables in Python
```

In [1]:
```
#Data exploration, here we assign the data, and then we visualize the data in a t

import pandas as pd
# Assign data   ye dictionary hain
data = {'Name': ['Neha', 'Samira', 'Prachiti', #string is to list  columns bhejte
                 'Anuja', 'Prasad', 'Poonam', 'Riya'],
        'Age': [17, 17, 18, 17, 18, 17, 17],
        'Gender': ['M', 'F', 'M', 'M', 'M', 'F', 'F'],
        'Marks': [90, 76, 'NaN', 74, 65, 'NaN', 71]}

# Convert into DataFrame
df = pd.DataFrame(data)

# Display data
df
```

Out[1]:

	Name	Age	Gender	Marks
0	Neha	17	M	90
1	Samira	17	F	76
2	Prachiti	18	M	NaN
3	Anuja	17	M	74
4	Prasad	18	M	65
5	Poonam	17	F	NaN
6	Riya	17	F	71

In [2]:
```
#Dealing with missing values

# Compute average

c = avg = 0
for ele in df['Marks']:
    if str(ele).isnumeric():
        c += 1
        avg += ele
avg /= c

# Replace missing values
df = df.replace(to_replace="NaN",  #NaN - Not a number
                value=avg)

# Display data  #in the fom of list
df
```

Out[2]:

	Name	Age	Gender	Marks
0	Neha	17	M	90.0
1	Samira	17	F	76.0
2	Prachiti	18	M	75.2
3	Anuja	17	M	74.0
4	Prasad	18	M	65.0
5	Poonam	17	F	75.2
6	Riya	17	F	71.0

In [3]:
```
#Reshaping data, in the GENDER column, we can reshape the data by categorizing th

# Categorize gender
df['Gender'] = df['Gender'].map({'M': 0,'F': 1, }).astype(float)

# Display data
df
```

Out[3]:

	Name	Age	Gender	Marks
0	Neha	17	0.0	90.0
1	Samira	17	1.0	76.0
2	Prachiti	18	0.0	75.2
3	Anuja	17	0.0	74.0
4	Prasad	18	0.0	65.0
5	Poonam	17	1.0	75.2
6	Riya	17	1.0	71.0

In [4]:
```
#Filtering data

# Filter top scoring students
df = df[df['Marks'] >= 75]

# Remove age column
df = df.drop(['Age'], axis=1) #axis=0 means row

# Display data
df
```

Out[4]:

	Name	Gender	Marks
0	Neha	0.0	90.0
1	Samira	1.0	76.0
2	Prachiti	0.0	75.2
5	Poonam	1.0	75.2

In [5]:
```
#Wrangling Data Using Merge Operation

#FIRST TYPE OF DATA:

# import module
import pandas as pd

# creating DataFrame for Student Details
details = pd.DataFrame({
'ID': [101, 102, 103, 104, 105, 106,107, 108, 109, 110],
'NAME': ['Jagroop', 'Praveen', 'Harjot','Pooja', 'Rahul',
         'Nikita','Saurabh', 'Ayush', 'Dolly', "Mohit"],
'BRANCH': ['CSE', 'CSE', 'CSE', 'CSE', 'CSE','CSE', 'CSE', 'CSE', 'CSE', 'CSE']})

# printing details
print(details)
```

```
    ID     NAME BRANCH
0  101  Jagroop    CSE
1  102  Praveen    CSE
2  103   Harjot    CSE
3  104    Pooja    CSE
4  105    Rahul    CSE
5  106   Nikita    CSE
6  107  Saurabh    CSE
7  108    Ayush    CSE
8  109    Dolly    CSE
9  110    Mohit    CSE
```

In [6]:
```
#SECOND TYPE OF DATA

# Import module
import pandas as pd

# Creating Dataframe for Fees_Status
fees_status = pd.DataFrame({'ID': [101, 102, 103, 104, 105,106, 107, 108, 109, 11
     'PENDING': ['5000', '250', 'NIL','9000', '15000', 'NIL','4500', '1800', '250

# Printing fees_status
print(fees_status)
```

```
    ID PENDING
0  101    5000
1  102     250
2  103     NIL
3  104    9000
4  105   15000
5  106     NIL
6  107    4500
7  108    1800
8  109     250
9  110     NIL
```

In [7]:
```
#WRANGLING DATA USING MERGE OPERATION:

# Import module
import pandas as pd

# Creating Dataframe
details = pd.DataFrame({'ID': [101, 102, 103, 104, 105,106, 107, 108, 109, 110],
'NAME': ['Jagroop', 'Praveen', 'Harjot','Pooja', 'Rahul', 'Nikita','Saurabh', 'Ay
'BRANCH': ['CSE', 'CSE', 'CSE', 'CSE', 'CSE','CSE', 'CSE', 'CSE', 'CSE', 'CSE']})

# Creating Dataframe
fees_status = pd.DataFrame({'ID': [101, 102, 103, 104, 105,106, 107, 108, 109, 11
'PENDING': ['5000', '250', 'NIL','9000', '15000', 'NIL','4500', '1800', '250', 'N

# Merging Dataframe
print(pd.merge(details, fees_status, on='ID'))# jo common column hain wo on="ID"
```

```
   ID     NAME BRANCH PENDING
0  101  Jagroop    CSE    5000
1  102  Praveen    CSE     250
2  103   Harjot    CSE     NIL
3  104    Pooja    CSE    9000
4  105    Rahul    CSE   15000
5  106   Nikita    CSE     NIL
6  107  Saurabh    CSE    4500
7  108    Ayush    CSE    1800
8  109    Dolly    CSE     250
9  110    Mohit    CSE     NIL
```

In [8]:
```
#CARS SELLING DATA

# Import module
import pandas as pd

# Creating Data
car_selling_data = {'Brand': ['Maruti', 'Maruti', 'Maruti','Maruti', 'Hyundai', '
                              'Mahindra', 'Mahindra','Ford', 'Toyota', 'Ford'],
'Year': [2010, 2011, 2009, 2013,2010, 2011, 2011, 2010,2013, 2010, 2010, 2011],
'Sold': [6, 7, 9, 8, 3, 5, 2, 8, 7, 2, 4, 2]}

# Creating Dataframe of car_selling_data
df = pd.DataFrame(car_selling_data)

# printing Dataframe
print(df)
```

```
       Brand  Year  Sold
0     Maruti  2010     6
1     Maruti  2011     7
2     Maruti  2009     9
3     Maruti  2013     8
4    Hyundai  2010     3
5    Hyundai  2011     5
6     Toyota  2011     2
7   Mahindra  2010     8
8   Mahindra  2013     7
9       Ford  2010     2
10    Toyota  2010     4
11      Ford  2011     2
```

In [33]:
```
#DATA OF THE YEAR 2010:

# Import mdoule
import pandas as pd

# Creating Data
car_selling_data = {'Brand': ['Maruti', 'Maruti', 'Maruti','Maruti', 'Hyundai', '
                              'Toyota', 'Mahindra', 'Mahindra','Ford', 'Toyota', 'F
'Year': [2010, 2011, 2009, 2013,2010, 2011, 2011, 2010,2013, 2010, 2010, 2011],
'Sold': [6, 7, 9, 8, 3, 5,2, 8, 7, 2, 4, 2]}

# Creating Dataframe for Provided Data
df = pd.DataFrame(car_selling_data)

# Group the data when year = 2010
grouped = df.groupby('Year')
print(grouped.get_group(2010))
```

```
       Brand  Year  Sold
0     Maruti  2010     6
4    Hyundai  2010     3
7   Mahindra  2010     8
9       Ford  2010     2
10    Toyota  2010     4
```

In [10]:

```
#DETAILS STUDENTS DATA WHO WANT TO PARTICIPATE IN THE EVENT:

# Import module
import pandas as pd
student_data = {'Name': ['Amit', 'Praveen', 'Jagroop','Rahul', 'Vishal', 'Suraj',
                         'Rahul', 'Praveen', 'Amit'],

'Roll_no': [23, 54, 29, 36, 59, 38,12, 45, 34, 36, 54, 23],

'Email': ['xxxx@gmail.com', 'xxxxxx@gmail.com','xxxxxx@gmail.com', 'xx@gmail.com'
                         'xxxx@gmail.com', 'xxxxx@gmail.com',
                         'xxxxx@gmail.com', 'xxxxx@gmail.com',
                         'xxxxx@gmail.com', 'xxxxxx@gmail.com',
                         'xxxxxxxxxx@gmail.com', 'xxxxxxxxxx@gmail.com']}

# Creating Dataframe of Data
df = pd.DataFrame(student_data)

# Printing Dataframe
print(df)
```

```
        Name  Roll_no                Email
0       Amit       23        xxxx@gmail.com
1    Praveen       54      xxxxxx@gmail.com
2    Jagroop       29      xxxxxx@gmail.com
3      Rahul       36          xx@gmail.com
4     Vishal       59        xxxx@gmail.com
5      Suraj       38       xxxxx@gmail.com
6     Rishab       12       xxxxx@gmail.com
7   Satyapal       45       xxxxx@gmail.com
8       Amit       34       xxxxx@gmail.com
9      Rahul       36      xxxxxx@gmail.com
10   Praveen       54  xxxxxxxxxx@gmail.com
11      Amit       23  xxxxxxxxxx@gmail.com
```

In [11]:
```
#DATA WRANGLED BY REMOVING DUPLICATE ENTRIES:

# import module
import pandas as pd
student_data = {'Name': ['Amit', 'Praveen', 'Jagroop','Rahul', 'Vishal', 'Suraj',
                         'Rishab', 'Satyapal', 'Amit',
                         'Rahul', 'Praveen', 'Amit'],

'Roll_no': [23, 54, 29, 36, 59, 38,12, 45, 34, 36, 54, 23],
'Email': ['xxxx@gmail.com', 'xxxxxx@gmail.com','xxxxxx@gmail.com', 'xx@gmail.com'
        'xxxx@gmail.com', 'xxxxx@gmail.com','xxxxx@gmail.com', 'xxxxx@gmail.com',
        'xxxxx@gmail.com', 'xxxxxx@gmail.com','xxxxxxxxxx@gmail.com', 'xxxxxxxxx
df = pd.DataFrame(student_data)
non_duplicate = df[~df.duplicated('Roll_no')]
print(non_duplicate)
```

```
       Name  Roll_no             Email
0      Amit       23    xxxx@gmail.com
1   Praveen       54  xxxxxx@gmail.com
2   Jagroop       29  xxxxxx@gmail.com
3     Rahul       36      xx@gmail.com
4    Vishal       59    xxxx@gmail.com
5     Suraj       38   xxxxx@gmail.com
6    Rishab       12   xxxxx@gmail.com
7  Satyapal       45   xxxxx@gmail.com
8      Amit       34   xxxxx@gmail.com
```

In [12]:
```
#Binning method for data smoothing

import numpy as np
import math
from sklearn.datasets import load_iris
from sklearn import datasets, linear_model, metrics
dataset = load_iris()
a = dataset.data
b = np.zeros(150)

for i in range (150):
    b[i]=a[i,1]

b=np.sort(b) #sort the array
bin1=np.zeros((30,5))
bin2=np.zeros((30,5))
bin3=np.zeros((30,5))
```

In [13]:
```
# Bin mean
for i in range (0,150,5):
    k=int(i/5)
    mean=(b[i] + b[i+1] + b[i+2] + b[i+3] + b[i+4])/5
    for j in range(5):
        bin1[k,j]=mean
print("Bin Mean: \n",bin1)
```

```
Bin Mean:
 [[2.18 2.18 2.18 2.18 2.18]
 [2.34 2.34 2.34 2.34 2.34]
 [2.48 2.48 2.48 2.48 2.48]
 [2.52 2.52 2.52 2.52 2.52]
 [2.62 2.62 2.62 2.62 2.62]
 [2.7  2.7  2.7  2.7  2.7 ]
 [2.74 2.74 2.74 2.74 2.74]
 [2.8  2.8  2.8  2.8  2.8 ]
 [2.8  2.8  2.8  2.8  2.8 ]
 [2.86 2.86 2.86 2.86 2.86]
 [2.9  2.9  2.9  2.9  2.9 ]
 [2.96 2.96 2.96 2.96 2.96]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.04 3.04 3.04 3.04 3.04]
 [3.1  3.1  3.1  3.1  3.1 ]
 [3.12 3.12 3.12 3.12 3.12]
 [3.2  3.2  3.2  3.2  3.2 ]
 [3.2  3.2  3.2  3.2  3.2 ]
 [3.26 3.26 3.26 3.26 3.26]
 [3.34 3.34 3.34 3.34 3.34]
 [3.4  3.4  3.4  3.4  3.4 ]
 [3.4  3.4  3.4  3.4  3.4 ]
 [3.5  3.5  3.5  3.5  3.5 ]
 [3.58 3.58 3.58 3.58 3.58]
 [3.74 3.74 3.74 3.74 3.74]
 [3.82 3.82 3.82 3.82 3.82]
 [4.12 4.12 4.12 4.12 4.12]]
```

In [14]:
```
# Bin boundaries
for i in range (0,150,5):
    k=int(i/5)
    for j in range (5):
        if (b[i+j]-b[i]) < (b[i+4]-b[i+j]):
            bin2[k,j]=b[i]
        else:
            bin2[k,j]=b[i+4]
print("Bin Boundaries: \n",bin2)
```

```
Bin Boundaries:
 [[2.  2.3 2.3 2.3 2.3]
 [2.3 2.3 2.3 2.4 2.4]
 [2.4 2.5 2.5 2.5 2.5]
 [2.5 2.5 2.5 2.5 2.6]
 [2.6 2.6 2.6 2.6 2.7]
 [2.7 2.7 2.7 2.7 2.7]
 [2.7 2.7 2.7 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.9 2.9 2.9]
 [2.9 2.9 2.9 2.9 2.9]
 [2.9 2.9 3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.1 3.1]
 [3.1 3.1 3.1 3.1 3.1]
 [3.1 3.1 3.1 3.1 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.3 3.3 3.3]
 [3.3 3.3 3.3 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.5 3.5 3.5 3.5 3.5]
 [3.5 3.6 3.6 3.6 3.6]
 [3.7 3.7 3.7 3.8 3.8]
 [3.8 3.8 3.8 3.8 3.9]
 [3.9 3.9 3.9 4.4 4.4]]
```

In [15]:
```
# Bin median
for i in range (0,150,5):
    k=int(i/5)
    for j in range (5):
        bin3[k,j]=b[i+2]
print("Bin Median: \n",bin3)
```

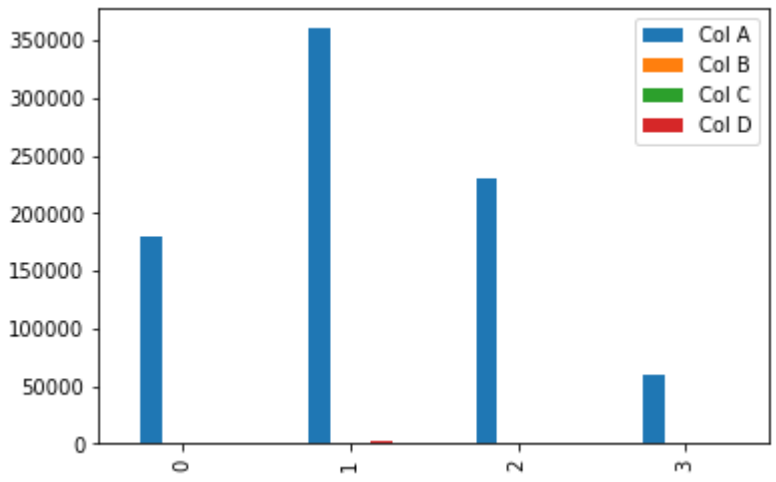
```
Bin Median:
 [[2.2 2.2 2.2 2.2 2.2]
 [2.3 2.3 2.3 2.3 2.3]
 [2.5 2.5 2.5 2.5 2.5]
 [2.5 2.5 2.5 2.5 2.5]
 [2.6 2.6 2.6 2.6 2.6]
 [2.7 2.7 2.7 2.7 2.7]
 [2.7 2.7 2.7 2.7 2.7]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.9 2.9 2.9 2.9 2.9]
 [2.9 2.9 2.9 2.9 2.9]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.1 3.1 3.1 3.1 3.1]
 [3.1 3.1 3.1 3.1 3.1]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.3 3.3 3.3 3.3 3.3]
 [3.3 3.3 3.3 3.3 3.3]
 [3.4 3.4 3.4 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.5 3.5 3.5 3.5 3.5]
 [3.6 3.6 3.6 3.6 3.6]
 [3.7 3.7 3.7 3.7 3.7]
 [3.8 3.8 3.8 3.8 3.8]
 [4.1 4.1 4.1 4.1 4.1]]
```

In [37]:
```
#Data Normalization with Pandas

# importing packages
import pandas as pd

# create data
df = pd.DataFrame([
                [180000, 110, 18.9, 1400],
                [360000, 905, 23.4, 1800],
                [230000, 230, 14.0, 1300],
                [60000, 450, 13.5, 1500]],

                columns=['Col A', 'Col B','Col C', 'Col D'])

# view data
display(df)
```

	Col A	Col B	Col C	Col D
0	180000	110	18.9	1400
1	360000	905	23.4	1800
2	230000	230	14.0	1300
3	60000	450	13.5	1500

In [17]:
```
import matplotlib.pyplot as plt
df.plot(kind = 'bar')
```

Out[17]: <AxesSubplot:>

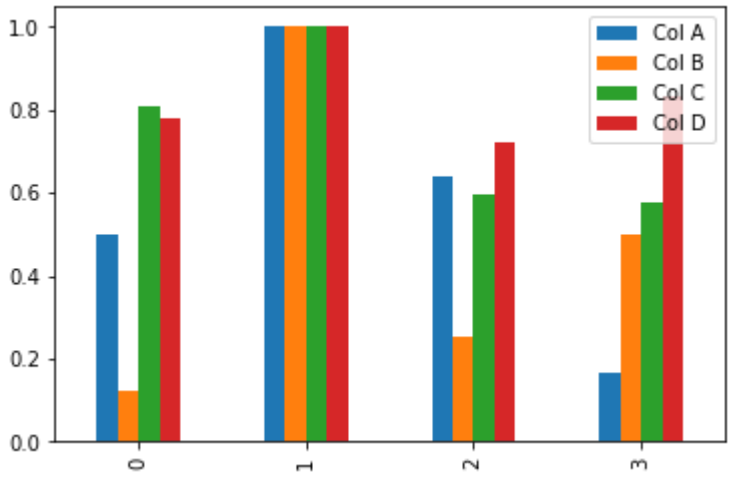


In [39]:
```
# copy the data
df_max_scaled = df.copy()

# apply normalization techniques
for column in df_max_scaled.columns:
    df_max_scaled[column] = df_max_scaled[column] / df_max_scaled[column].abs().m

# view normalized data
display(df_max_scaled)
```

	Col A	Col B	Col C	Col D
0	0.500000	0.121547	0.807692	0.777778
1	1.000000	1.000000	1.000000	1.000000
2	0.638889	0.254144	0.598291	0.722222
3	0.166667	0.497238	0.576923	0.833333

In [19]:
```
import matplotlib.pyplot as plt
df_max_scaled.plot(kind = 'bar')
```

Out[19]: <AxesSubplot:>

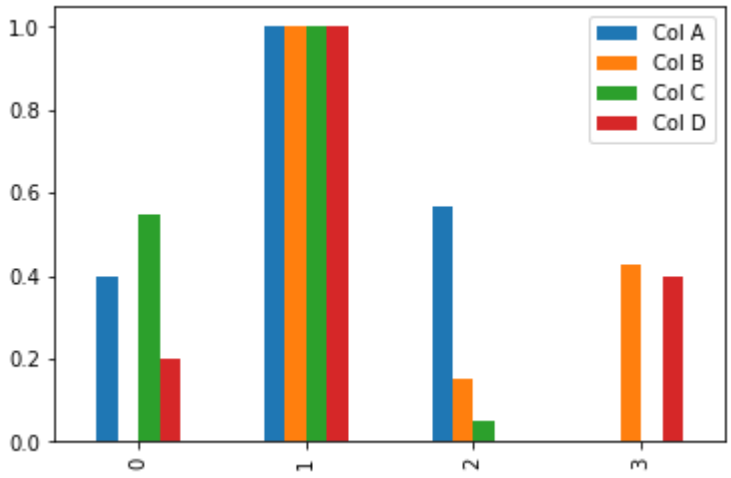


In [20]:
```
# copy the data
df_min_max_scaled = df.copy()

# apply normalization techniques
for column in df_min_max_scaled.columns:
    df_min_max_scaled[column] = (df_min_max_scaled[column] - df_min_max_scaled[co

# view normalized data
print(df_min_max_scaled)
```

```
      Col A     Col B     Col C  Col D
0  0.400000  0.000000  0.545455    0.2
1  1.000000  1.000000  1.000000    1.0
2  0.566667  0.150943  0.050505    0.0
3  0.000000  0.427673  0.000000    0.4
```

In [21]:
```
import matplotlib.pyplot as plt
df_min_max_scaled.plot(kind = 'bar')
```

Out[21]: <AxesSubplot:>

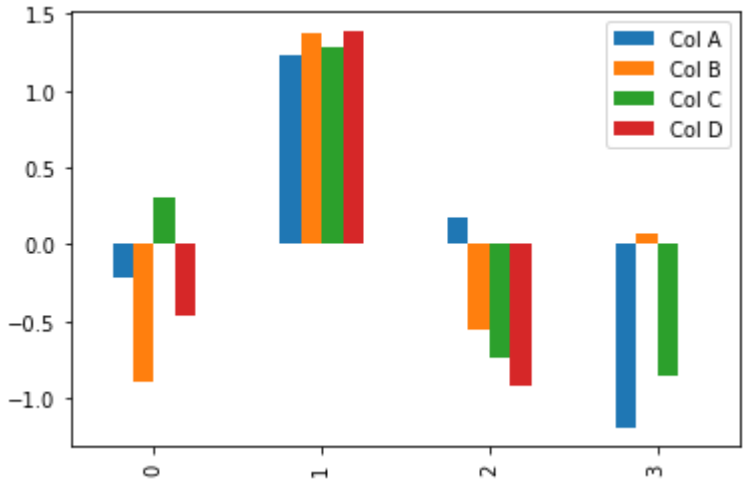


In [22]:
```
# copy the data
df_z_scaled = df.copy()

# apply normalization techniques
for column in df_z_scaled.columns:
    df_z_scaled[column] = (df_z_scaled[column] -
                           df_z_scaled[column].mean()) / df_z_scaled[column].std()

# view normalized data
display(df_z_scaled)
```

	Col A	Col B	Col C	Col D
0	-0.221422	-0.895492	0.311486	-0.46291
1	1.227884	1.373564	1.278167	1.38873
2	0.181163	-0.552993	-0.741122	-0.92582
3	-1.187625	0.074922	-0.848531	0.00000

In [23]:
```
import matplotlib.pyplot as plt
df_z_scaled.plot(kind='bar')
```

Out[23]: <AxesSubplot:>



In [ ]: