Aggregations and Descriptive Statistics in Panda

सीबीएसई पाठ्यक्रम पर आधारित

इन्फार्मेटिक्स प्रैक्टिसेज कक्षा -12

अध्याय-3

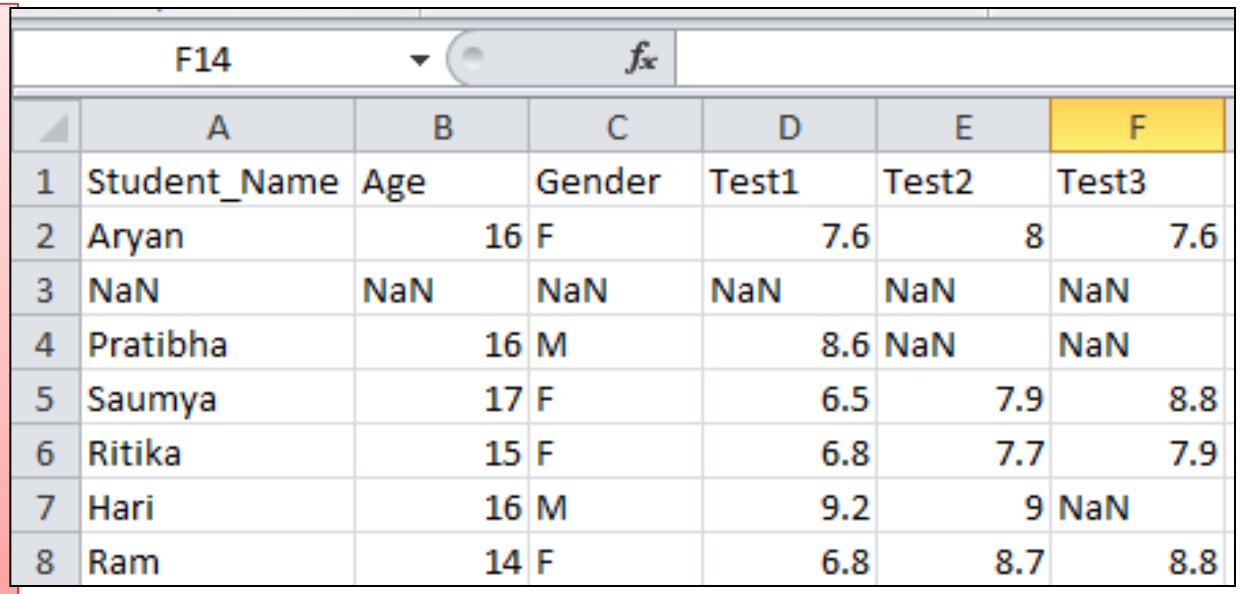
द्वारा:
संजीव भदौरिया
स्नातकोत्तर शिक्षक (संगणक विज्ञान)
के० वि० बाराबंकी (लखनऊ संभाग)

परिचय

- Data analysis के लिए पाइथन एक बहुत अच्छी कंप्यूटर भाषा है |
- Python pandas इस प्रकार के data को analyze करने के लिए कई प्रकार के data aggregation function प्रदान करता है |
- Analysis का एक महत्वपूर्ण कार्य होता है बड़े dataset का summarization जिसे aggregations को compute करना कहते हैं जैसे- sum(), mean(), median(), min() और max() जिसमे विशाल dataset से एक संख्या मिल जाती है |
- इस अध्याय में हम इन्ही data aggregate functions का प्रयोग करना सीखेंगे |
- Aggregation का अर्थ होता है एक विशाल dataset की values को process करके एक value प्राप्त करना |
- Data aggregation में हमेशा multivalued functions दिए जाते हैं जो एक single value return करते हैं |
- दिया जाने वाला dataset या तो series या DataFrame होता है |

Data Aggregation Functions

- हम निम्न excel sheet से एक dataframe बनाते हैं – जिसका process निम्न है|



F14

	A	B	C	D	E	F
1	Student_Name	Age	Gender	Test1	Test2	Test3
2	Aryan	16	F	7.6	8	7.6
3	NaN	NaN	NaN	NaN	NaN	NaN
4	Pratibha	16	M	8.6	NaN	NaN
5	Saumya	17	F	6.5	7.9	8.8
6	Ritika	15	F	6.8	7.7	7.9
7	Hari	16	M	9.2	9	NaN
8	Ram	14	F	6.8	8.7	8.8

```
>>> import pandas as pd
>>> df=pd.read_csv("C:\\Users\\KVBBKServer\\Desktop\\Student.csv")
>>> df
  Student_Name   Age Gender  Test1  Test2  Test3
0        Aryan  16.0      F    7.6    8.0    7.6
1          NaN   NaN    NaN    NaN    NaN    NaN
2     Pratibha  16.0      M    8.6    NaN    NaN
3       Saumya  17.0      F    6.5    7.9    8.8
4       Ritika  15.0      F    6.8    7.7    7.9
5         Hari  16.0      M    9.2    9.0    NaN
6          Ram  14.0      F    6.8    8.7    8.8
```

Data Aggregation Functions

DataFrame "df" में प्रत्येक column में प्रत्येक row के लिए उपस्थित non-NaN items को गिनकर प्रदर्शित करना |

```
>>> df.count()
Student_Name      6
Age               6
Gender            6
Test1             6
Test2             5
Test3             4
dtype: int64
```

उपरोक्त उदहारण में आपने देखा की प्रत्येक column में बिना NaN वाली values को ही गिना गया है |

DataFrame "df" में से किसी एक column के लिए उपस्थित non-NaN items को गिनकर प्रदर्शित करना हो तो निम्न दो तरीकों में से कोई भी तरीका अपना सकते हैं |

```
>>> df.Age.count()
6
>>> df['Age'].count()
6
>>> df.Test3.count()
4
>>> df['Test3'].count()
4
```

DataFrame के column सभी non-NaN values को जोड़ने के लिए निम्न तरीका अपनाया जा सकता है |

```
>>> df.sum()
Age          94.0
Test1        45.5
Test2        41.3
Test3        33.1
dtype: float64
```

DataFrame के row के सभी non-NaN values को जोड़ने के लिए निम्न तरीका अपनाया जा सकता है |

```
>>> df.sum(axis=1)
0    39.2
1     0.0
2    24.6
3    40.2
4    37.4
5    34.2
6    38.3
```

Data Aggregation Functions

```
>>> S=pd.Series([5,10,15,20,25])
>>> S.count()
5
>>> S.sum()
75
>>> S.mean()
15.0
>>> S.median()
15.0
```

```
>>> S.max()
25
>>> S.min()
5
>>> S.std()
7.905694150420948
>>> S.var()
62.5
```

```
>>> S=pd.Series([5,10,15,10,10,10,25])
>>> S.mode()
0    10
dtype: int64
```

Aggregation	Description
count()	कुल items की संख्या
sum()	दिए गए संख्याओं के सेट के items का sum को हल करता है |
mean()	दिए गए संख्याओं के सेट के items का mean या average को हल करता है |
median()	दिए गए संख्याओं के सेट के items के बीच की संख्या या median को हल करता है |
mode()	दिए गए संख्याओं के सेट के items में सबसे ज्यादा repeated value या mode को हल करता है |
max()	दिए गए संख्याओं के सेट में से सबसे बड़ी संख्या को खोजता है |
min()	दिए गए संख्याओं के सेट में से सबसे छोटी संख्या को खोजता है |
std()	दिए गए संख्याओं के सेट के items का standared deviation calculate करता है |
var()	दिए गए संख्याओं के सेट के items का variance calculate करता है |

Data Aggregation Functions

```
>>> df.min()
Age       14.0
Test1      6.5
Test2      7.7
Test3      7.6
dtype: float64
>>> df.Age.min()
14.0
>>> df.Age.std()
1.0327955589886446
```

```
>>> df.var()
Age       1.066667
Test1     1.209667
Test2     0.313000
Test3     0.382500
dtype: float64
>>> df.Age.var()
1.0666666666666669
```

Assignment:

State wise sales values of an item is given below-

State	Sales
Goa	650000
Delhi	692400
Odisha	750000
Haryana	867000
Bihar	920000
Kerala	939000
Tamil Nadu	1015000
West Bengal	1553000
Maharashtra	2176000

Write Commands for the following (Dataframe name is dfA)

(a) Count the number of observation in dfA.
(b) Count the number of observation in column state of dfA.
(c) Sum of the non-null value across the row axis for dfA.
(d) Calculate the mean of Sales columns in dfA.
(e) Add a commission column into dfA. (Commission = sales*0.04)
(f) Calculate the mean of all numeric columns in dfA.
(g) Calculate the mean of sales column.
(h) Find maximum sales and commission values.
(i) Find minimum sales and commission values.
(j) Find the standard deviation of commissions.

Quantiles with Pandas

- Statistics में 3 शब्द बहुत प्रयोग में लाये जाते हैं - Quartile, Quantile और percentile. इनमे निम्न अंतर होता है |
  - 0 quartile = 0 quantile = 0 percentile
  - 1 quartile = 0.25 quantile = 25 percentile
  - 2 quartile = 0.50 quantile = 50 percentile
  - 3 quartile = 0.75 quantile = 75 percentile
  - 4 quartile = 1 quantile = 100 percentile
- Statistics में ***Quartile*** आपके data को 4 हिस्सों (quarter) में बाँट देता है |
- ***Percentile*** एक संख्या है, जहां एक निश्चित percentage स्कोर, उस संख्या से नीचे आता है | इनका प्रयोग अधिकतर परीक्षा इत्यादि में स्कोर को रिपोर्ट करने में किया जाता है |
  - Percentile = ((N-your Rank)/N)*100
  - Your rank = (percentile/100) * number of items
- Quantiles एक distribution में वह point हैं जो उस distribution में values के rank order से संबंधित हैं |

Pandas में Quantile () का प्रयोग

- Pandas का quantile() function किसी request किये गए axis के लिए float अथवा series values को return करता है | request किया गया axis एक numpy percentile होता है | यह एक probabilities की list होती है जिस पर quantile compute किया जाना है |
  - माना यदि percentile = 25 है तो यह पहला quartile अथवा lower quartile होगा |
  - यदि percentile = 50 है तो यह दूसरा quartile अथवा median होगा |
  - यदि percentile = 75 है तो यह तीसरा quartile अथवा upper quartile होगा |
- इसका syntax निम्न है –

  DataFrame.quantile(q=0.5, axis=0, numeric_only=True, interpolation = 'linear')
- जहाँ -
  - q एक float है या array के जैसा, इसका default 0.5 (50% quantile) और 0<=q<=1
  - axis : 0 अथवा 'index' और 1 अथवा 'columns' | इसका default 0 होता है |
  - numeric_only : boolean, default True होता है |
  - interpolation: {'linear', 'lower', 'higher', 'midpoint', 'nearest'}. यह वैकल्पिक (optional) होता है |

Pandas में Quantile () का प्रयोग

Find the Quantile of an odd series s given bellow:

```
>>> s=pd.Series([15,16,18,27,29,32,36])
>>> s.quantile(.25)
17.0
>>> s.quantile([0.25,0.5,0.75])
0.25    17.0
0.50    27.0
0.75    30.5
dtype: float64
```

आप निम्न कमांड भी try कर सकते हैं |

```
>>> import numpy as np
>>> S=pd.Series([15,16,18,27,29,32,36]
>>> np.percentile(S,25)
17.0
```

Find the Quantile of an even series s given bellow:

```
>>> P=pd.Series([15,16,18,29,32,36])
>>> P.quantile([.25,.5,.75])
0.25    16.50
0.50    23.50
0.75    31.25
dtype: float64
```

आप निम्न कमांड भी try कर सकते हैं |

```
>>> import numpy as np
>>> S=pd.Series([15,16,18,29,32,36])
>>> print(np.percentile(P,[25,50,75]))
[16.5  23.5  31.25]
```

Pandas में Quantile () का प्रयोग

Find the quantile of DataFrame "df"

```
>>> import pandas as pd
>>> df=pd.read_csv("C:\\Users\\KVBBKServer\\Desktop\\Student.csv")
>>> df
  Student_Name   Age Gender  Test1  Test2  Test3
0        Aryan  16.0      F    7.6    8.0    7.6
1          NaN   NaN    NaN    NaN    NaN    NaN
2      Pratibha  16.0      M    8.6    NaN    NaN
3       Saumya  17.0      F    6.5    7.9    8.8
4       Ritika  15.0      F    6.8    7.7    7.9
5         Hari  16.0      M    9.2    9.0    NaN
6          Ram  14.0      F    6.8    8.7    8.8
>>> df.quantile(.25)
Age      15.250
Test1     6.800
Test2     7.900
Test3     7.825
Name: 0.25, dtype: float64
>>> df.quantile([.25,.5,.75])
        Age  Test1  Test2  Test3
0.25  15.25   6.80    7.9  7.825
0.50  16.00   7.20    8.0  8.350
0.75  16.00   8.35    8.7  8.800
```

```
>>> quants=[0.05,0.25,0.5,0.75,0.95]
>>> q=df.quantile(quants)
>>> print(q)
        Age  Test1  Test2  Test3
0.05  14.25  6.575   7.74  7.645
0.25  15.25  6.800   7.90  7.825
0.50  16.00  7.200   8.00  8.350
0.75  16.00  8.350   8.70  8.800
0.95  16.75  9.050   8.94  8.800
```

Descriptive Statistics

- Python – pandas में descriptive या summary statistics के लिए ***describe ( )*** function का प्रयोग किया जाता है |
- Describe ( ) के द्वारा mean , std और interquartile (IQR) values को हासिल किया जा सकता है |
- Describe ( ) एक numeric columns पर काम करने के लिए बहुत ही आसन function होता है |
- किसी column के basic statistics को देखे के लिए आप describe ( ) function का प्रयोग कर सकते हैं –जैसे - mean, min, max इत्यादि |
- सामान्यतया describe ( ) character columns को छोड़ देता है और सिर्फ numeric columns पर कार्य करता है |
- किसी dataframe को describe करने पर सिर्फ numeric fields ही return होती है | और यह 8 प्रकार की statistical properties को दर्शाता है -

count( )	mean()
std()	min()
25^{th} percentile	50^{th} percentile
75^{th} percentile	max()

Descriptive Statistics

- Describe ( ) function का syntax निम्नवत है -

DataFrame.describe(percentile = None, include = None, exclude=None)

जहाँ : -

- percentile : default है [0.25, 0.5, 0.75]
- Include: default None है, अन्य में 'All' होसकता है |
- Exclude : भी default None है , यह तब प्रयोग में लाया जाता है जब आप किसी भी column को इन्क्लुड़े न करना चाहें |

```
>>> S=pd.Series([5,10,15,20,25])
>>> S.describe()
count      5.000000
mean      15.000000
std        7.905694
min        5.000000
25%       10.000000
50%       15.000000
75%       20.000000
max       25.000000
dtype: float64
```

STRING टाइप के series के लिए |

```
>>> str=pd.Series(['a','a','b','b','c','d'])
>>> str.describe()
count     6
unique    4
top       a
freq      2
dtype: object
```

Descriptive Statistics

```
>>> import pandas as pd
>>> df=pd.read_csv("C:\\Users\\KVBBKServer\\Desktop\\Student.csv")
>>> df
  Student_Name   Age Gender  Test1  Test2  Test3
0        Aryan  16.0      F    7.6    8.0    7.6
1          NaN   NaN    NaN    NaN    NaN    NaN
2     Pratibha  16.0      M    8.6    NaN    NaN
3       Saumya  17.0      F    6.5    7.9    8.8
4       Ritika  15.0      F    6.8    7.7    7.9
5         Hari  16.0      M    9.2    9.0    NaN
6          Ram  14.0      F    6.8    8.7    8.8
>>> df.describe()
             Age      Test1      Test2      Test3
count   6.000000   6.000000   5.000000   4.000000
mean   15.666667   7.583333   8.260000   8.275000
std     1.032796   1.099848   0.559464   0.618466
min    14.000000   6.500000   7.700000   7.600000
25%    15.250000   6.800000   7.900000   7.825000
50%    16.000000   7.200000   8.000000   8.350000
75%    16.000000   8.350000   8.700000   8.800000
max    17.000000   9.200000   9.000000   8.800000
```

DataFrame के लिए describe( ) function.

```
>>> df['Age'].describe()
count      6.000000
mean      15.666667
std        1.032796
min       14.000000
25%       15.250000
50%       16.000000
75%       16.000000
max       17.000000
Name: Age, dtype: float64
```

Descriptive Statistics (Assignment)

Example A dataframe dfS is given with following data for 10 students:

	Name	Total Marks
0	Amit Kumar	450
1	Vinamr Katyal	476
2	Aasha Goel	426
3	Naina Rawat	476
4	Dawn Sebastian	458
5	Riya Mary	446
6	Aashna Sharma	461
7	Piush Cocher	464
8	Om Berma	476
9	Manali Sovani	452

Write commands for the following:

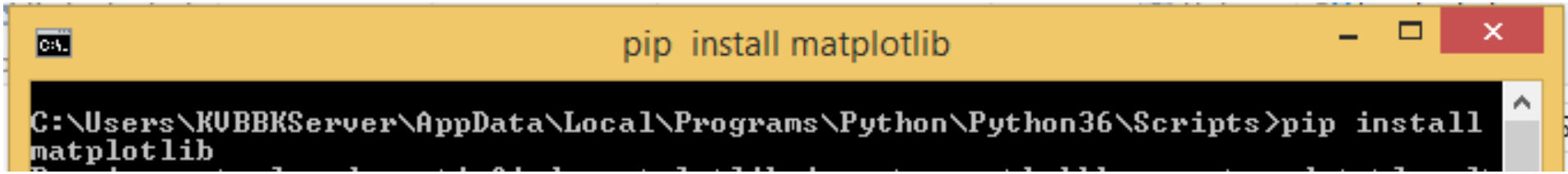
(*a*) Find the default quantile of the DataFrame.
(*b*) Find the [.25, .5, .75] quantiles of the DataFrame.
(*c*) Write the summary of statistics pertaining to the dataframe column.
(*d*) Get the full summary of statistics to the dataframe.
(*e*) Find the 50th percentile of the dataframe.

Histogram

- Histogram किसी data के distribution को analyze करने के लिए एक powerful टूल है |
- एक histogram plot को सामान्यतया किसी संख्या की frequency को दर्शाने के लिए प्रयोग किया जाता है |
- इसके द्वारा user को data का विभिन्न catagory में distribution आसानी से समझ में आजाता है | तथा साथ ही data की median और range भी समझ आजाती है |
- Histogram बनाने के लिए, पहले, हम values की पूरी range को intervals की एक series में विभाजित करते हैं। और दूसरा, हम गिनते हैं कि प्रत्येक interval में कितने values आते हैं|
- Matplotlib फिर bins में उन categories या intervals को call करता है | bins, variables के continuous और non-overlapping intervals होते हैं। वे एक दूसरे के ठीक लगे हुए (adjacent) और बराबर size के होने चाहिए|
- Histogram में -
  - **X – axis**: observation के intervals को दर्शाता है |
  - **Y- axis:** यह frequency के घनत्व (density) को दर्शाता है |

Matplotlib

- Matplotlib पाइथन की एक अग्रणी visualization library है जो कि एक powerful और two dimentional plotting library है |
- यह numpy arrays के आधार पर बनी हुई एक multi-platform data visualization library है |
- यह सभी प्रकार के graph, plots, charts, histograms इत्यादि बनाने में सक्षम है |
- इसके लिए आपको अपने system में pip कमांड के द्वारा matplotlib library को install करना होता है |




```
>>> import matplotlib
>>> print(matplotlib.__version__)
3.0.1
```

इस कमांड से हम पता कर सकते हैं की matplotlib का कौन सा version installed है |

Histogram बनाना

- Histogram बनाने के लिए syntax निम्न है -

```
DataFrame.hist(column=None, by=None, grid=True, xlabelsize=None, xrot=None, ylabelsize=None, yrot=None, ax=None, sharex=False, sharey=False, figsize=None, layout=None, bins=10, **kwds)
```

- column: is the dataframe column name to create a histogram.
- by: If passed, then used to form histograms for separate groups. The by option will take an object by which the data can be grouped.
- grid: takes a Boolean value, i.e., to enable (if **True**) or disable (if **False**) the grid.
- xlabelsize, ylabelsize: these options change the size of x and y label text size.
- sharex, sharey: to set both of the axes to the same range and scale.
- bin: isumber of histogram bins to be used. The default value is 10.
- fill: to fillis the dataframe column name to create a histogram.

- Example के लिए हम निम्न DataFrame लेते हैं -

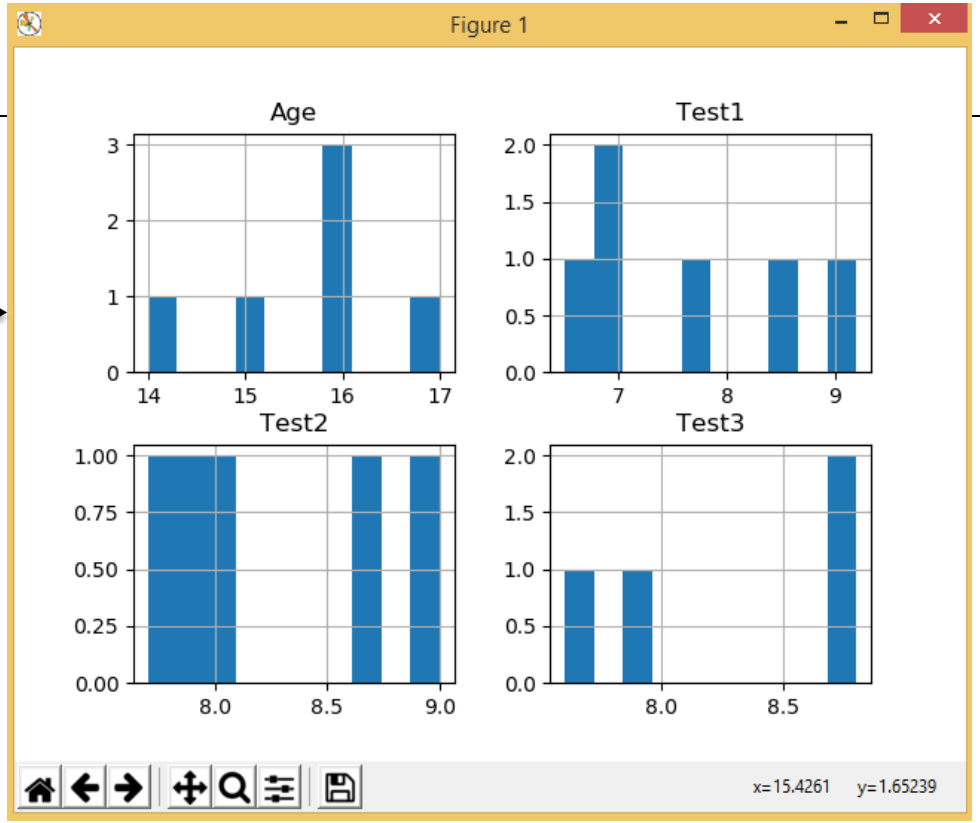
```
>>> df
  Student_Name   Age Gender  Test1  Test2  Test3
0        Aryan  16.0      F    7.6    8.0    7.6
1          NaN   NaN    NaN    NaN    NaN    NaN
2     Pratibha  16.0      M    8.6    NaN    NaN
3       Saumya  17.0      F    6.5    7.9    8.8
4       Ritika  15.0      F    6.8    7.7    7.9
5         Hari  16.0      M    9.2    9.0    NaN
6          Ram  14.0      F    6.8    8.7    8.8
```

Histogram बनाना

```
>>> import matplotlib.pyplot as plt
>>> df.hist()
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000009B7FCB0080>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000009B7FD1A7F0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000009B7FE83D68>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000009B7FEB3320>]],
      dtype=object)
>>> plt.show()
```

Histogram बनाने के लिए hist( ) function का प्रयोग करते हैं |

उपरोक्त कमांड चलाने पर बगल में दर्शाई गयी छवि जो histogram बन कर आरहा है वही output आपको प्रदर्शित होता है |

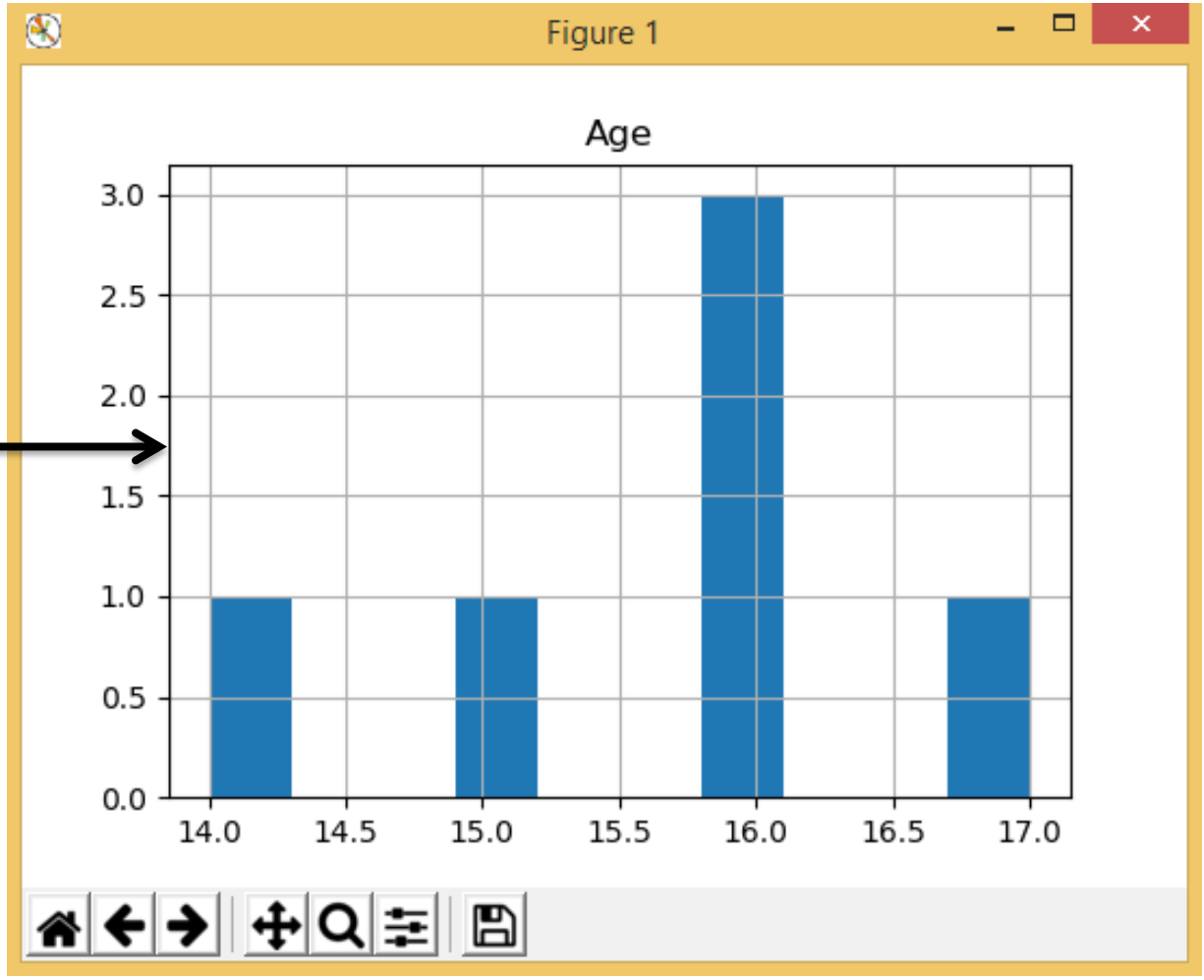


Pandas dataframe से single Histogram बनाना

hist ( ) function में column pass कर देते हैं |

```
>>> df.hist(column='Age')
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000009B7FBB2630>]],
      dtype=object)
>>> plt.show()
```

उपरोक्त कमांड चलाने पर बगल में दर्शाई गयी छवि जो histogram बन कर आरहा है वही output आपको प्रदर्शित होता है |

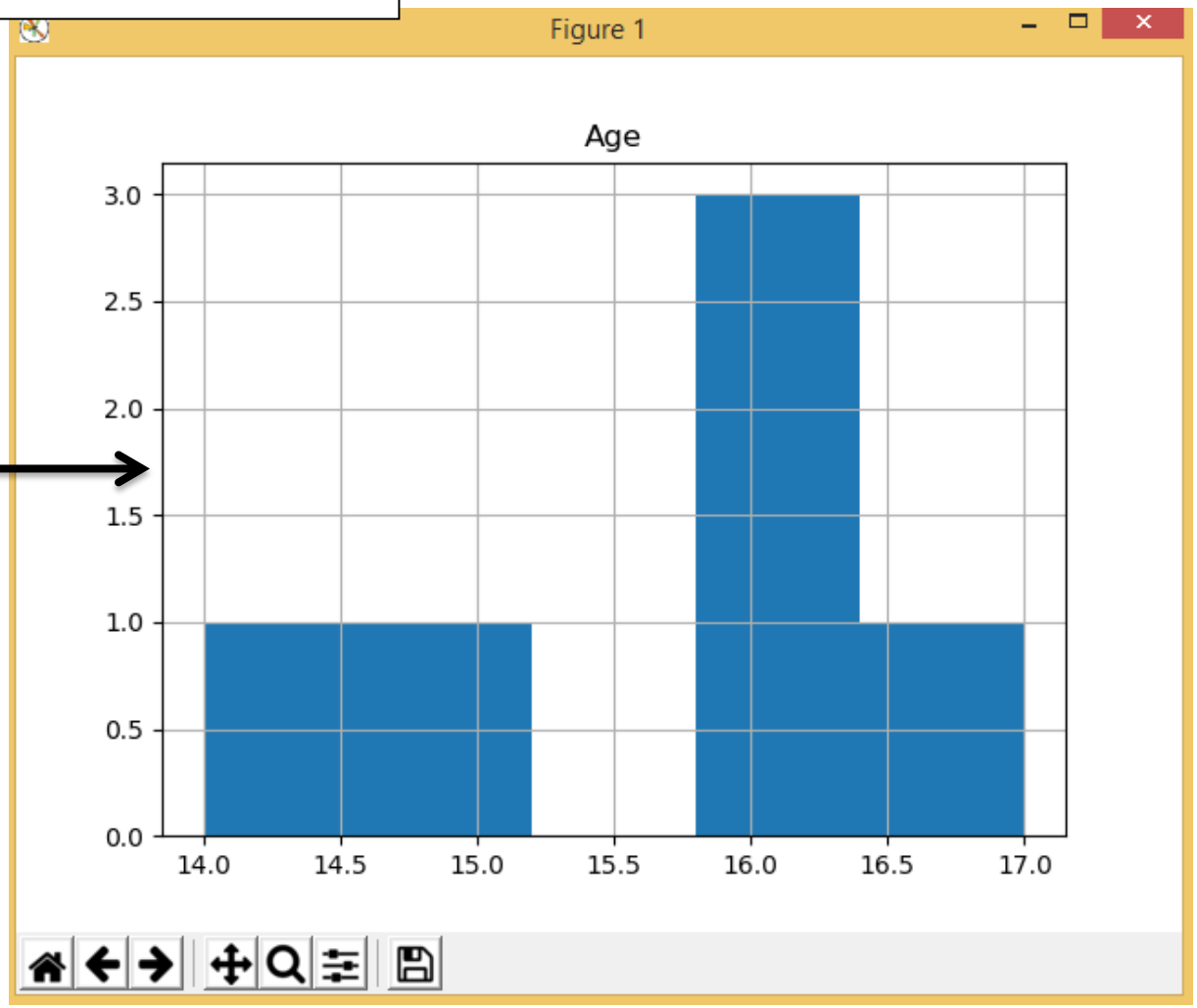


Histogram के bins के आकार को बदलना

```
>>> df.hist(column='Age', bins=5)
array([[<matplotlib.axes._subplot
       dtype=object)
>>> plt.show()
```

hist ( ) function में column के साथ bins को भी pass कर देते हैं |

उपरोक्त कमांड चलाने पर बगल में दर्शाई गयी छवि जो histogram बन कर आरहा है वही output आपको प्रदर्शित होता है |

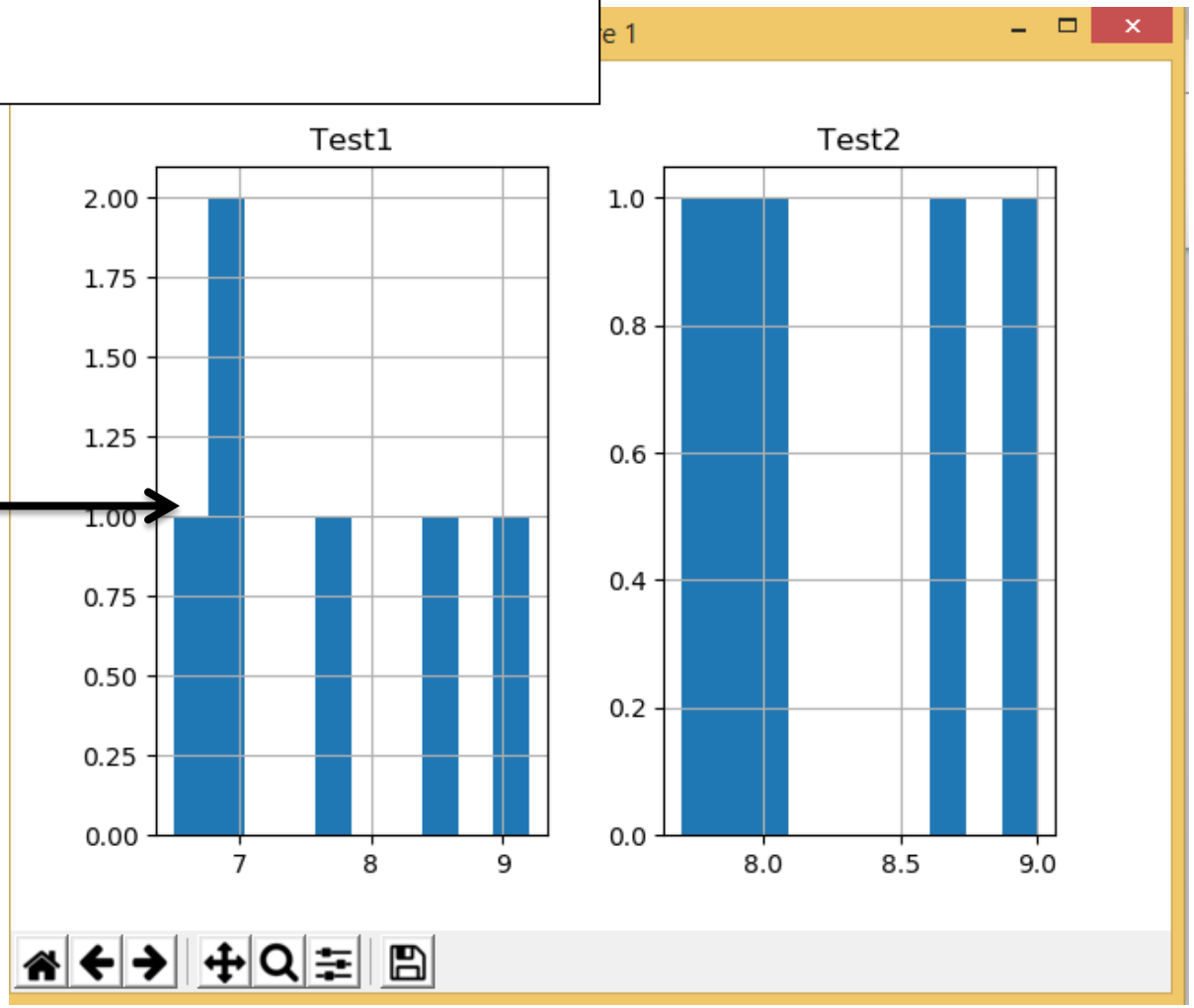


Multiple pandas Histogram

```
>>> df.hist(column=["Test1","Test2"])
array([[<matplotlib.axes._subplots.Axe
        <matplotlib.axes._subplots.Axe
       dtype=object)
>>> plt.show()
```

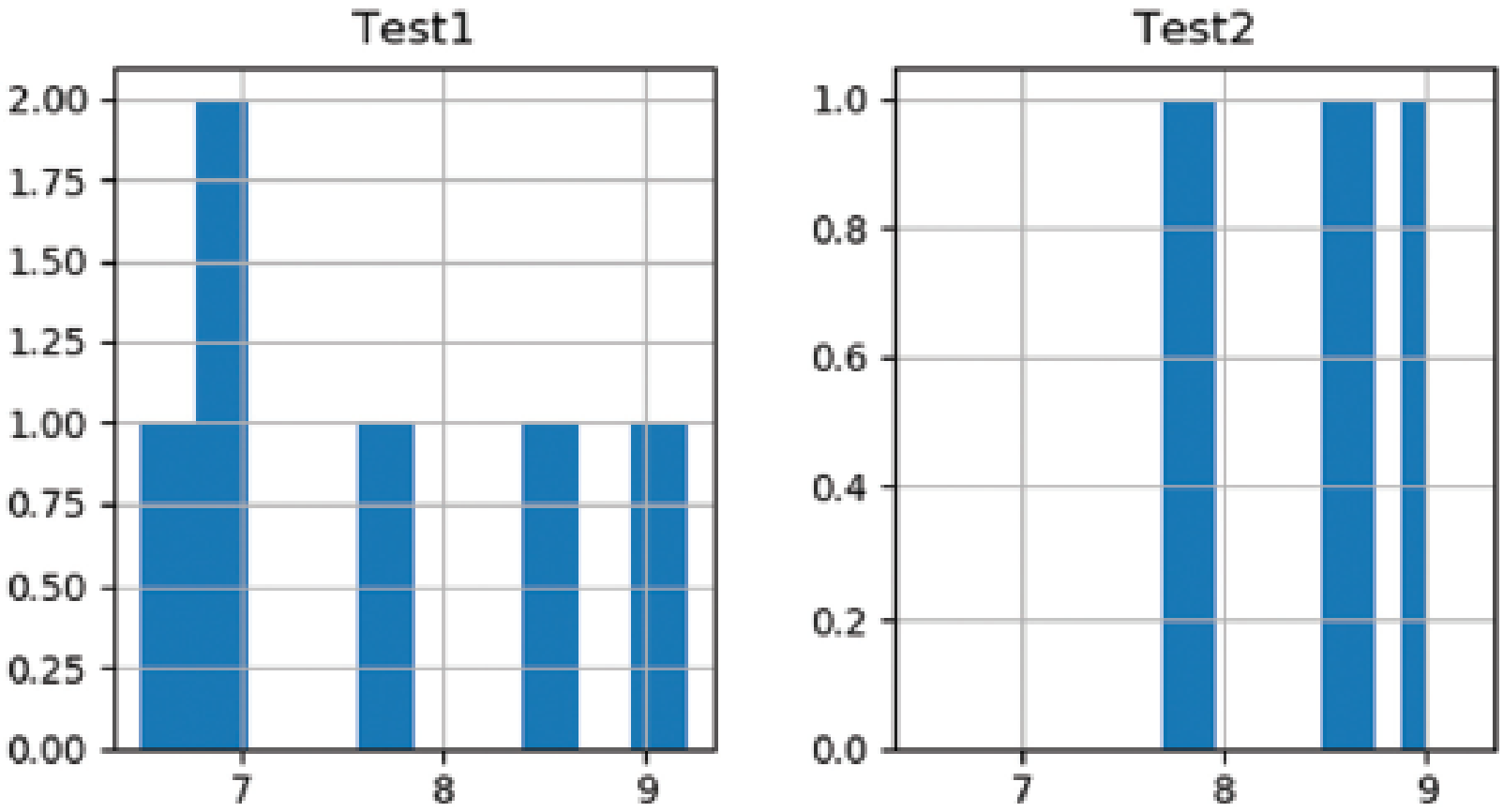
hist ( ) function में column को भी pass कर देते हैं |

उपरोक्त कमांड चलाने पर बगल में दर्शाई गयी छवि जो histogram बन कर आरहा है वही output आपको प्रदर्शित होता है |



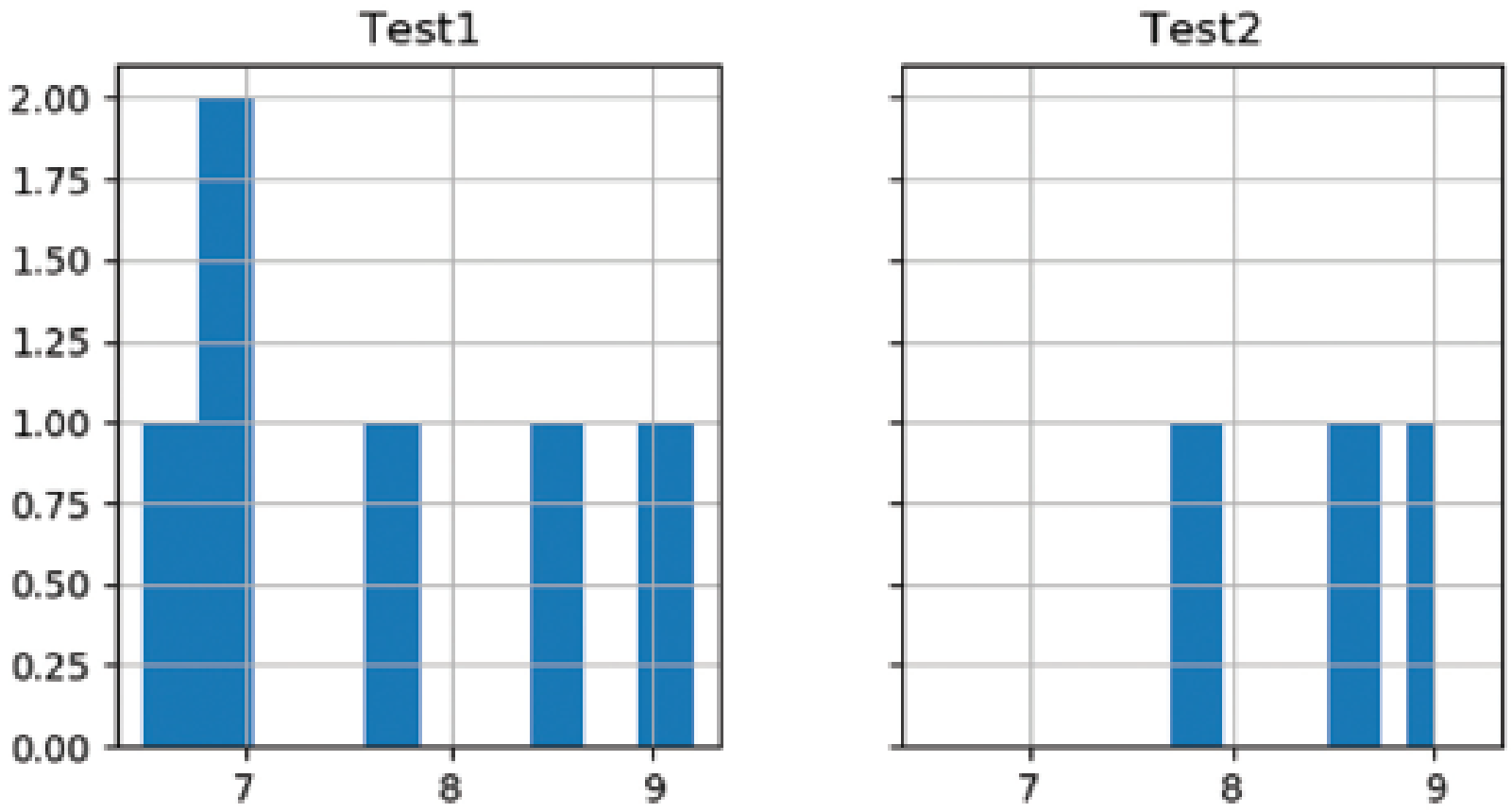
Histogram Axes को बदलना

```
>>> df.hist(column=["Test1", "Test2"], sharex=True)  # Share only x axis
>>> plt.show()
```



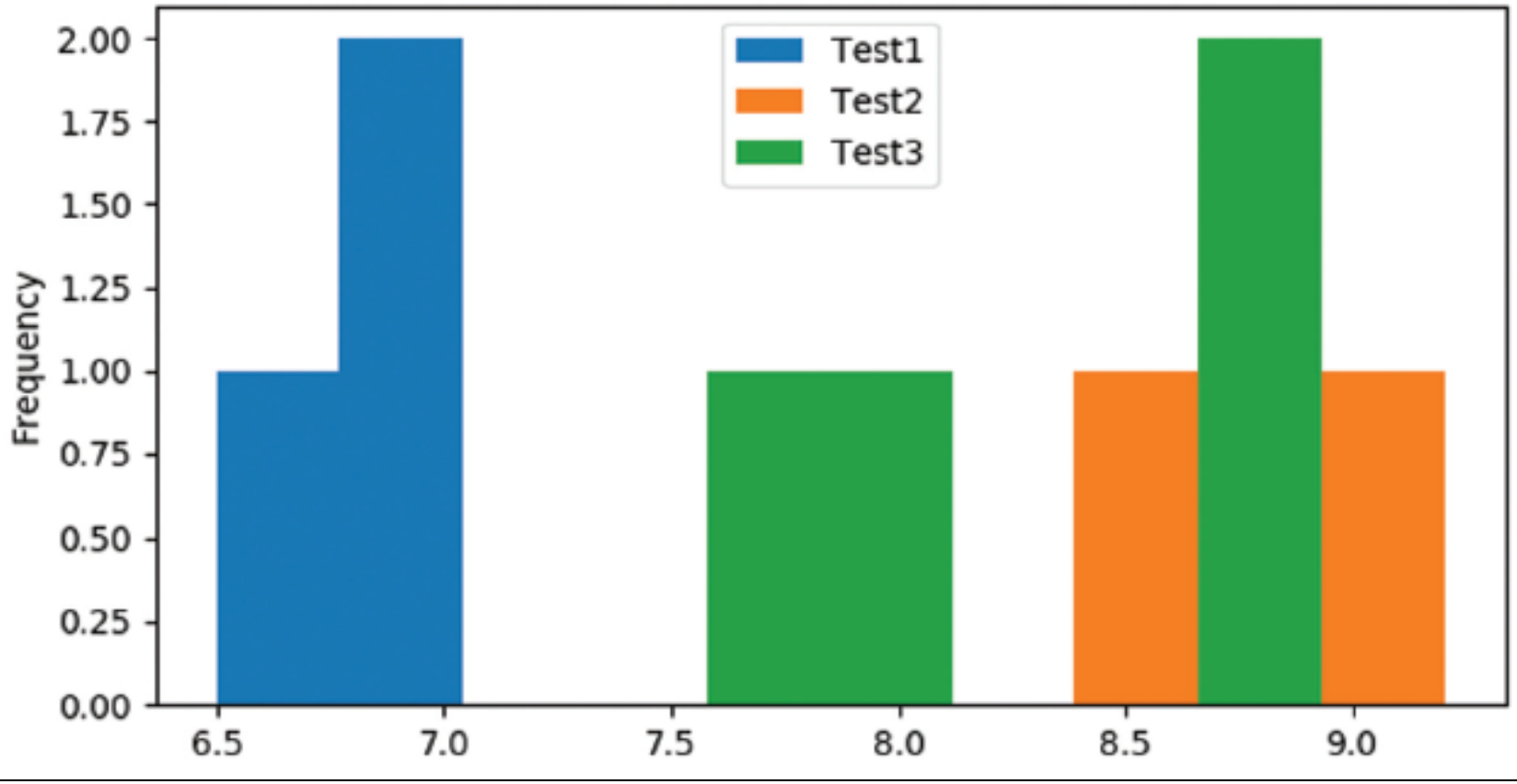
Histogram Axes को बदलना

```
>>> df.hist(column=["Test1", "Test2"], sharex=True, sharey=True)  # Share x and y axis
>>> plt.show()
```



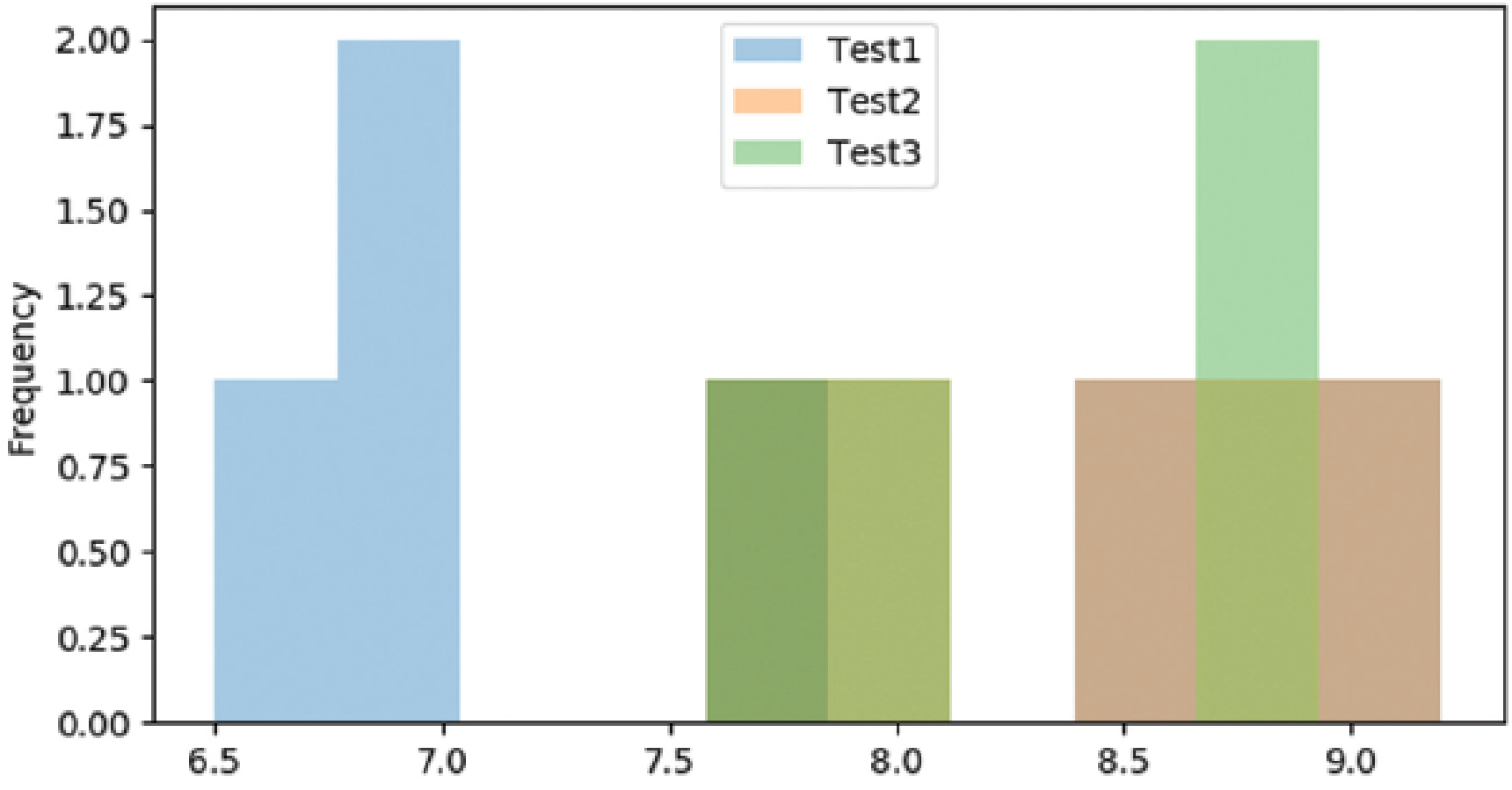
Multiple features in one plot

```
>>> df[["Test1", "Test2", "Test3"]].plot.hist()  # Note slicing is performed on df itself
>>> plt.show()
```



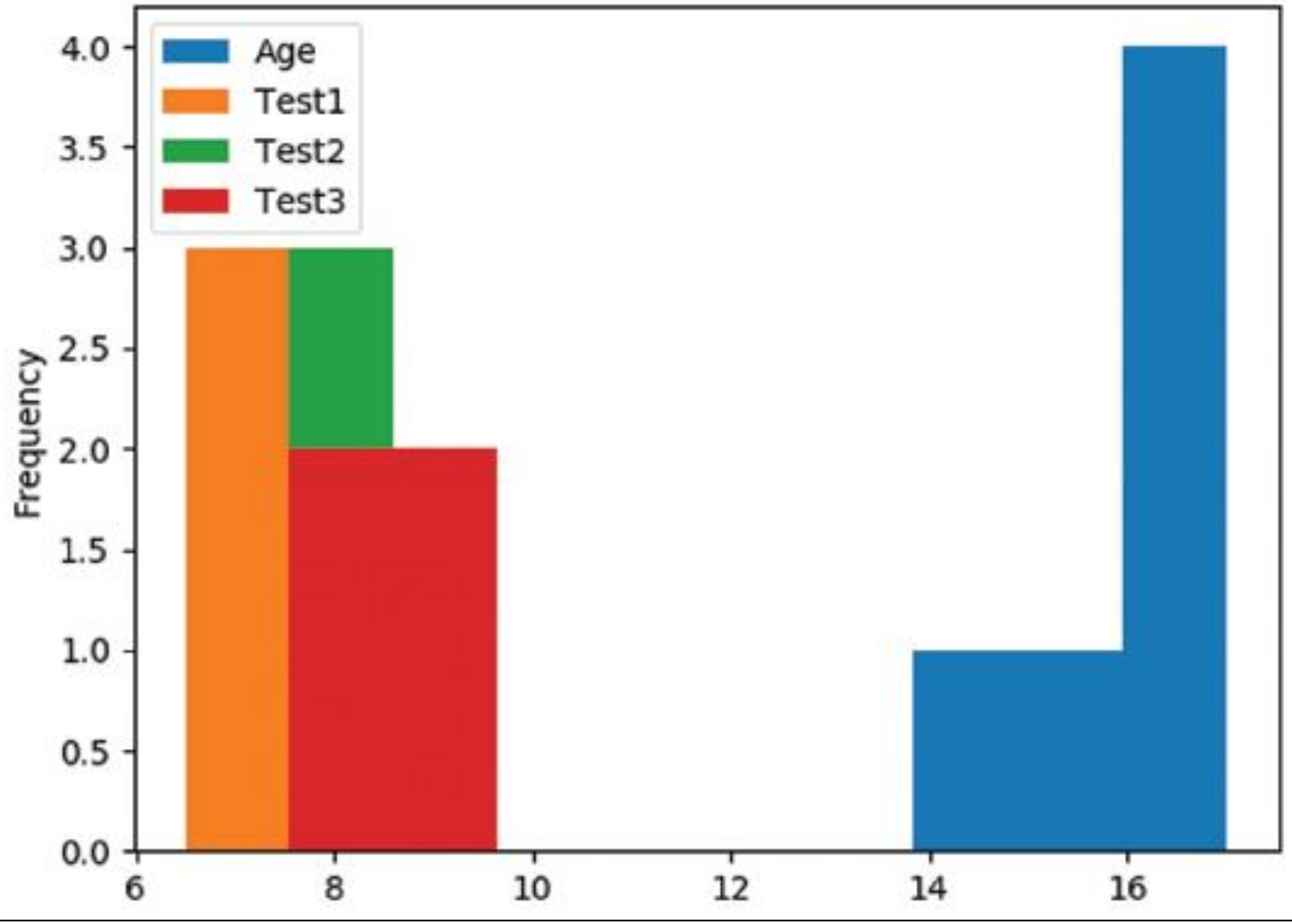
Multiple features in one plot

```
>>> df[["Test1", "Test2", "Test3"]].plot.hist(alpha=0.4)  # Plot at 40% opacity
>>> plt.show()
```



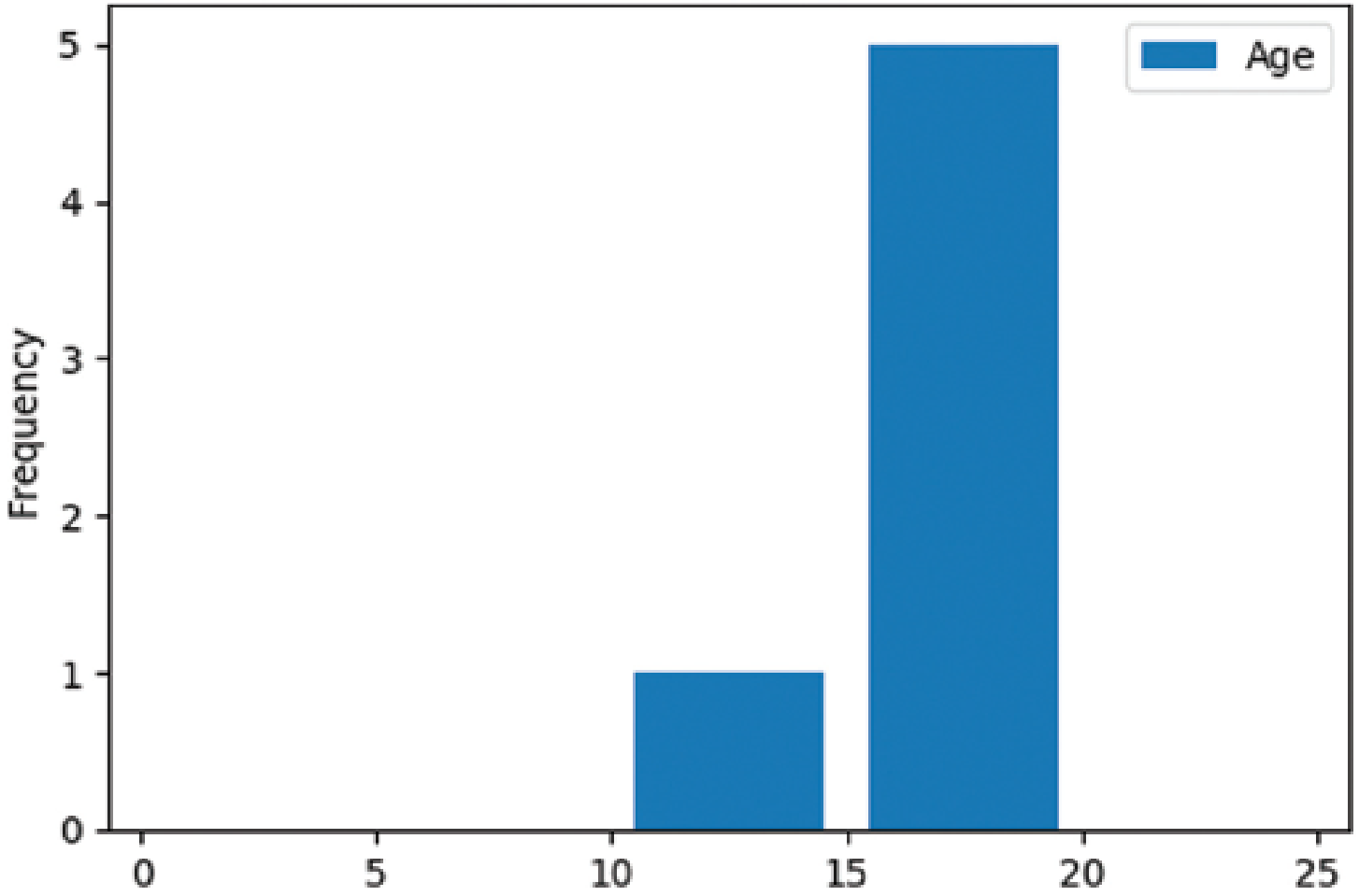
Plotting DataFrame Columns using DataFrame plot () Method

```
>>> df.plot(kind='hist')
>>> plt.show()
```



Histogram using single/multiple column

```
>>> df[['Age']].plot(kind='hist',bins=[0,5,10,15,20, 25],rwidth=0.8)
>>> plt.show()
```



Assignment

Example **Using previous dataframe dfS, write the command to create following histograms:**

(*a*) Crete a histogram plot to show Total Marks

(*b*) Create a histogram plot to show Total Marks with bins 100.

(*c*) Create a histogram plot to show Total Marks with bins [400,420,440,460,480,500].

Solution For data: dfA = pd.read_csv('E:/IPSource_XII/IPXIIChap03/Std7.csv')

Assume that the following modules are imported.

```
import pandas as pd
import matplotlib.pyplot as plt
```

(*a*)
```
dfA = dfA.sort_values(by='Total Marks')
    plt.show()
```

(*b*)
```
dfA.hist(column="Total Marks", bins=200)
    plt.show()
```

(*c*)
```
dfA.hist(column="Total Marks", bins=[400,420,440,460,480,500])
    plt.show()
    Or
    dfA[['Total Marks']].plot(kind='hist',bins=[400,420,440,460,480,500])
    plt.show()
```

- कृपया हमारे ब्लॉग को फॉलो करिए और youtube channel को subscribe करिए | ताकि आपको और सारे chapters मिल सकें |

www.pythontrends.wordpress.com