pandas_lab

April 11, 2020

[1]:
```
import pandas as pd # import pandas module as pd
```

[2]:
```
# create a series with a list
s = pd.Series([1,4,-2,'home'],index=['a','b','c','d'])
```

[3]:
```
print(s)
```

```
a       1
b       4
c      -2
d    home
dtype: object
```

[4]:
```
#Todo 1
print(s)
#according to thos datatype of s is object
```

```
a       1
b       4
c      -2
d    home
dtype: object
```

[5]:
```
print(s['d'])
```

```
home
```

[6]:
```
print(s['a'])
```

```
1
```

[7]:
```
s.astype(int)
```

```
␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)
```

```
      <ipython-input-7-ee92bad5c5cf> in <module>
   ----> 1 s.astype(int)


      c:
↪\users\user\appdata\local\programs\python\python38-32\lib\site-packages\pandas\core\generic.
↪py in astype(self, dtype, copy, errors)
      5696         else:
      5697             # else, only a single dtype is given
   -> 5698             new_data = self._data.astype(dtype=dtype, copy=copy,␣
↪errors=errors)
      5699             return self._constructor(new_data).__finalize__(self)
      5700


      c:
↪\users\user\appdata\local\programs\python\python38-32\lib\site-packages\pandas\core\internal
↪py in astype(self, dtype, copy, errors)
       580
       581     def astype(self, dtype, copy: bool = False, errors: str =␣
↪"raise"):
   --> 582         return self.apply("astype", dtype=dtype, copy=copy,␣
↪errors=errors)
       583
       584     def convert(self, **kwargs):


      c:
↪\users\user\appdata\local\programs\python\python38-32\lib\site-packages\pandas\core\internal
↪py in apply(self, f, filter, **kwargs)
       440                 applied = b.apply(f, **kwargs)
       441             else:
   --> 442                 applied = getattr(b, f)(**kwargs)
       443             result_blocks = _extend_blocks(applied, result_blocks)
       444


      c:
↪\users\user\appdata\local\programs\python\python38-32\lib\site-packages\pandas\core\internal
↪py in astype(self, dtype, copy, errors)
       623             vals1d = values.ravel()
       624             try:
   --> 625                 values = astype_nansafe(vals1d, dtype, copy=True)
       626             except (ValueError, TypeError):
       627                 # e.g. astype_nansafe can fail on object-dtype of␣
↪strings
```

```
        c:
↪\users\user\appdata\local\programs\python\python38-32\lib\site-packages\pandas\core\dtypes\c
↪py in astype_nansafe(arr, dtype, copy, skipna)
        872         # work around NumPy brokenness, #1987
        873         if np.issubdtype(dtype.type, np.integer):
    --> 874             return lib.astype_intsafe(arr.ravel(), dtype).
↪reshape(arr.shape)
        875
        876         # if we have a datetime/timedelta array of objects


        pandas\_libs\lib.pyx in pandas._libs.lib.astype_intsafe()


        ValueError: invalid literal for int() with base 10: 'home'
```

[8]:
```
print(s.astype(str))
```

```
a       1
b       4
c      -2
d    home
dtype: object
```

[9]:
```
print(type(s['a']))
```

```
<class 'int'>
```

[10]:
```
print(type(s['d']))
```

```
<class 'str'>
```

[11]:
```
#so according to above results
#datatype of s cannot be chaged --- answer for the Todo 1
```

[12]:
```
# create a data frame with a dictionary
data={'population':[1.5,1.2,2.0,1.4,0.8],
      'state':['Nevada','Florida','O hio','Texas','Florida'],
      'year':[2003,2000,2004,1990,1994]
     }
```

[14]:
```
print("data = ",data)
```

```
data =  {'population': [1.5, 1.2, 2.0, 1.4, 0.8], 'state': ['Nevada', 'Florida',
'O hio', 'Texas', 'Florida'], 'year': [2003, 2000, 2004, 1990, 1994]}
```

[15]:
```
df=pd.DataFrame(data,index=['one','two','three','four','five'],columns=␣
↪['year','state','population','debt'])
```

[17]:
```
print("df = \n",df)
```

```
df =
        year    state  population debt
one     2003   Nevada         1.5  NaN
two     2000  Florida         1.2  NaN
three   2004    O hio         2.0  NaN
four    1990    Texas         1.4  NaN
five    1994  Florida         0.8  NaN
```

[18]:
```
#4.2.2 Accessing and modifying
```

[19]:
```
print(s[1:3])
```

```
b     4
c    -2
dtype: object
```

[20]:
```
print(s[0])
```

```
1
```

[21]:
```
print(s['d'])
```

```
home
```

[22]:
```
print(s.values[2:])
```

```
[-2 'home']
```

[23]:
```
print(df[['population','state']])
```

```
       population    state
one           1.5   Nevada
two           1.2  Florida
three         2.0    O hio
four          1.4    Texas
five          0.8  Florida
```

[24]:
```
print(df.population)
```

```
one      1.5
two      1.2
three    2.0
four     1.4
```

```
five     0.8
Name: population, dtype: float64
```

[25]:
```
print(df.iloc[1:])
```

```
       year    state  population debt
two    2000  Florida         1.2  NaN
three  2004    O hio         2.0  NaN
four   1990    Texas         1.4  NaN
five   1994  Florida         0.8  NaN
```

[26]:
```
print(df.iloc[2:4:,2:5])
```

```
       population debt
three         2.0  NaN
four          1.4  NaN
```

[27]:
```
print(df.loc['one'])
```

```
year            2003
state         Nevada
population       1.5
debt             NaN
Name: one, dtype: object
```

[28]:
```
df.debt=34.67
```

[29]:
```
print("df = \n",df)
```

```
df =
        year    state  population   debt
one    2003   Nevada         1.5  34.67
two    2000  Florida         1.2  34.67
three  2004    O hio         2.0  34.67
four   1990    Texas         1.4  34.67
five   1994  Florida         0.8  34.67
```

[30]:
```
df.debt=[df.iloc[:,2][i]*5 for i in range(0,df.shape[0])]
```

[31]:
```
print("df = \n",df)
```

```
df =
        year    state  population  debt
one    2003   Nevada         1.5   7.5
two    2000  Florida         1.2   6.0
three  2004    O hio         2.0  10.0
four   1990    Texas         1.4   7.0
five   1994  Florida         0.8   4.0
```

[32]:
```
print("df = \n",df.head())
```

```
df =
        year    state  population  debt
one     2003   Nevada         1.5   7.5
two     2000  Florida         1.2   6.0
three   2004    O hio         2.0  10.0
four    1990    Texas         1.4   7.0
five    1994  Florida         0.8   4.0
```

[33]:
```
print(df.tail(2))
```

```
      year    state  population  debt
four  1990    Texas         1.4   7.0
five  1994  Florida         0.8   4.0
```

[34]:
```
print(df.sample(n=3))
```

```
      year    state  population  debt
one   2003   Nevada         1.5   7.5
two   2000  Florida         1.2   6.0
five  1994  Florida         0.8   4.0
```

[35]:
```
print(df.sample(n=3))
```

```
       year    state  population  debt
two    2000  Florida         1.2   6.0
five   1994  Florida         0.8   4.0
three  2004    O hio         2.0  10.0
```

[36]:
```
import numpy as np

df['newColomn']=pd.Series(np.random.randn(df.shape[0]),index=df.index)
```

[37]:
```
print("df = \n",df)
```

```
df =
        year    state  population  debt  newColomn
one     2003   Nevada         1.5   7.5   0.789634
two     2000  Florida         1.2   6.0   0.538847
three   2004    O hio         2.0  10.0  -1.840678
four    1990    Texas         1.4   7.0  -0.603043
five    1994  Florida         0.8   4.0   0.963335
```

[38]:
```
print(df.drop_duplicates('state'))
```

```
       year    state  population  debt  newColomn
one    2003   Nevada         1.5   7.5   0.789634
```

```
two    2000  Florida         1.2   6.0   0.538847
three  2004    O hio         2.0  10.0  -1.840678
four   1990    Texas         1.4   7.0  -0.603043
```

[39]:
```
print(df.state)
```

```
one       Nevada
two      Florida
three      O hio
four       Texas
five     Florida
Name: state, dtype: object
```

[40]:
```
#4.2.3 Loading data from CSV file
```

[41]:
```
# without setting names
df=pd.read_csv('E:/University Works/3rd Year/Semester 6/CO 544 - Machine␣
 ↪Learning and Data Mining/Lab/2/sampleDataSet.csv')
```

[42]:
```
print("df = \n",df)
```

```
df =
     5.1  0.222222222       3.5     0.625       1.4  0.06779661   0.2  \
0    4.9     0.166667  3.000000  0.416667  1.400000    0.067797  0.20
1    4.7     0.111111  3.200000  0.500000  1.300000         NaN  0.20
2    4.6     0.083333  3.100000  0.458333  1.500000    0.084746  0.20
3    NaN     0.194444  3.600000  0.666667  1.400000         NaN  0.20
4    NaN     0.305556  3.900000  0.791667  1.700000    0.118644  0.40
..   …          …       …        …        …          …    …
94   7.2     0.805556  3.000000  0.416667  5.800000    0.813559  1.60
95   7.4          NaN  0.333333  6.100000  0.864407    1.900000  0.75
96   7.9     0.999900  3.800000  0.750000  6.400000    0.915254  2.00
97   6.4     0.583333  2.800000  0.333333  5.600000    0.779661  2.20
98   6.3     0.555556  2.800000  0.333333  5.100000    0.694915  1.50

    0.041666667     setosa
0    0.041666667     setosa
1    0.041666667     setosa
2    0.041666667     setosa
3    0.041666667     setosa
4          0.125     setosa
..           …        …
94         0.625  virginica
95    virginica        NaN
96   0.791666667  virginica
97         0.875  virginica
98   0.583333333  virginica
```

```
[99 rows x 9 columns]
```

[43]:
```
# setting names
df=pd.read_csv('E:/University Works/3rd Year/Semester 6/CO 544 - Machine Learning and Data Mining/Lab/2/sampleDataSet.csv',names=['a','b','c','d','e','f','g','h','i'])
```

[44]:
```
print("df = \n",df)
```

```
df =
       a         b         c         d         e         f     g            h  \
0    5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20  0.041666667
1    4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20  0.041666667
2    4.7  0.111111  3.200000  0.500000  1.300000       NaN  0.20  0.041666667
3    4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20  0.041666667
4    NaN  0.194444  3.600000  0.666667  1.400000       NaN  0.20  0.041666667
..   …       …         …         …         …         …     …           …
95   7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60       0.625
96   7.4       NaN  0.333333  6.100000  0.864407  1.900000  0.75   virginica
97   7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00  0.791666667
98   6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20       0.875
99   6.3  0.555556  2.800000  0.333333  5.100000  0.694915  1.50  0.583333333

             i
0       setosa
1       setosa
2       setosa
3       setosa
4       setosa
..         …
95   virginica
96         NaN
97   virginica
98   virginica
99   virginica

[100 rows x 9 columns]
```

[45]:
```
#Todo 2
#without names take the first row of the sampleDataSet as the names of the columns
#with names it take the names as the names of each column in the sampleDataSet
```

[46]:
```
#4.2.4 Dealing with missing values.
```

[47]:
```
print(df.isnull().g)
```

```
0     False
```

```
1     False
2     False
3     False
4     False
      …
95    False
96    False
97    False
98    False
99    False
Name: g, Length: 100, dtype: bool
```

[48]:
```
print(df.isnull().sum(0))
```

```
a    4
b    1
c    0
d    3
e    2
f    2
g    1
h    1
i    1
dtype: int64
```

[49]:
```
df=df[df.isnull().a != True]
```

[51]:
```
print("df = \n",df)
```

```
df =
       a         b         c         d         e         f     g            h  \
0    5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20  0.041666667
1    4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20  0.041666667
2    4.7  0.111111  3.200000  0.500000  1.300000       NaN  0.20  0.041666667
3    4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20  0.041666667
7    5.0  0.194444  3.400000       NaN  1.500000  0.084746  0.20  0.041666667
..   …       …         …         …         …         …     …            …
95   7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60        0.625
96   7.4       NaN  0.333333  6.100000  0.864407  1.900000  0.75    virginica
97   7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00  0.791666667
98   6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20        0.875
99   6.3  0.555556  2.800000  0.333333  5.100000  0.694915  1.50  0.583333333

           i
0     setosa
1     setosa
2     setosa
3     setosa
```

```
7      setosa
..        …
95  virginica
96        NaN
97  virginica
98  virginica
99  virginica

[96 rows x 9 columns]
```

[52]:
```
print(df.dropna(axis=0).isnull().sum())
```

```
a    0
b    0
c    0
d    0
e    0
f    0
g    0
h    0
i    0
dtype: int64
```

[53]:
```
print(df.dropna(axis=1))
```

```
      a         c
0   5.1  3.500000
1   4.9  3.000000
2   4.7  3.200000
3   4.6  3.100000
7   5.0  3.400000
..  …       …
95  7.2  3.000000
96  7.4  0.333333
97  7.9  3.800000
98  6.4  2.800000
99  6.3  2.800000

[96 rows x 2 columns]
```

[54]:
```
print(df.dropna(axis=1, how='all'))
```

```
      a         b         c         d         e         f     g            h  \
0   5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20  0.041666667
1   4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20  0.041666667
2   4.7  0.111111  3.200000  0.500000  1.300000       NaN  0.20  0.041666667
3   4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20  0.041666667
7   5.0  0.194444  3.400000       NaN  1.500000  0.084746  0.20  0.041666667
```

```
..  …       …       …       …       …       …       …           …
95  7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60        0.625
96  7.4       NaN  0.333333  6.100000  0.864407  1.900000  0.75    virginica
97  7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00  0.791666667
98  6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20        0.875
99  6.3  0.555556  2.800000  0.333333  5.100000  0.694915  1.50  0.583333333

            i
0      setosa
1      setosa
2      setosa
3      setosa
7      setosa
..        …
95  virginica
96        NaN
97  virginica
98  virginica
99  virginica

[96 rows x 9 columns]
```

[55]:
```
print(df.dropna(axis=1, thresh=1))
```

```
      a         b         c         d         e         f     g             h  \
0   5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20  0.041666667
1   4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20  0.041666667
2   4.7  0.111111  3.200000  0.500000  1.300000       NaN  0.20  0.041666667
3   4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20  0.041666667
7   5.0  0.194444  3.400000       NaN  1.500000  0.084746  0.20  0.041666667
..  …       …       …       …       …       …     …           …
95  7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60        0.625
96  7.4       NaN  0.333333  6.100000  0.864407  1.900000  0.75    virginica
97  7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00  0.791666667
98  6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20        0.875
99  6.3  0.555556  2.800000  0.333333  5.100000  0.694915  1.50  0.583333333

            i
0      setosa
1      setosa
2      setosa
3      setosa
7      setosa
..        …
95  virginica
96        NaN
97  virginica
98  virginica
```

```
99  virginica

[96 rows x 9 columns]
```

[56]:
```
print(df.drop('i',axis=1))
```

```
      a         b         c         d         e         f     g            h
0   5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20  0.041666667
1   4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20  0.041666667
2   4.7  0.111111  3.200000  0.500000  1.300000       NaN  0.20  0.041666667
3   4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20  0.041666667
7   5.0  0.194444  3.400000       NaN  1.500000  0.084746  0.20  0.041666667
..  …       …       …       …       …       …   …           …
95  7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60        0.625
96  7.4       NaN  0.333333  6.100000  0.864407  1.900000  0.75    virginica
97  7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00  0.791666667
98  6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20        0.875
99  6.3  0.555556  2.800000  0.333333  5.100000  0.694915  1.50  0.583333333

[96 rows x 8 columns]
```

[57]:
```
print(df.fillna(899))
```

```
      a           b         c           d         e           f     g  \
0   5.1    0.222222  3.500000    0.625000  1.400000    0.067797  0.20
1   4.9    0.166667  3.000000    0.416667  1.400000    0.067797  0.20
2   4.7    0.111111  3.200000    0.500000  1.300000  899.000000  0.20
3   4.6    0.083333  3.100000    0.458333  1.500000    0.084746  0.20
7   5.0    0.194444  3.400000  899.000000  1.500000    0.084746  0.20
..  …         …       …         …       …         …   …
95  7.2    0.805556  3.000000    0.416667  5.800000    0.813559  1.60
96  7.4  899.000000  0.333333    6.100000  0.864407    1.900000  0.75
97  7.9    0.999900  3.800000    0.750000  6.400000    0.915254  2.00
98  6.4    0.583333  2.800000    0.333333  5.600000    0.779661  2.20
99  6.3    0.555556  2.800000    0.333333  5.100000    0.694915  1.50

              h          i
0   0.041666667     setosa
1   0.041666667     setosa
2   0.041666667     setosa
3   0.041666667     setosa
7   0.041666667     setosa
..          …       …
95        0.625  virginica
96    virginica        899
97  0.791666667  virginica
98        0.875  virginica
99  0.583333333  virginica
```

```
[96 rows x 9 columns]
```

[58]:
```
print(df.fillna(method='ffill'))
```

```
      a         b         c         d         e         f     g            h  \
0   5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20  0.041666667
1   4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20  0.041666667
2   4.7  0.111111  3.200000  0.500000  1.300000  0.067797  0.20  0.041666667
3   4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20  0.041666667
7   5.0  0.194444  3.400000  0.458333  1.500000  0.084746  0.20  0.041666667
..  …       …         …         …         …         …     …           …
95  7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60       0.625
96  7.4  0.805556  0.333333  6.100000  0.864407  1.900000  0.75   virginica
97  7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00  0.791666667
98  6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20       0.875
99  6.3  0.555556  2.800000  0.333333  5.100000  0.694915  1.50  0.583333333

            i
0      setosa
1      setosa
2      setosa
3      setosa
7      setosa
..        …
95  virginica
96  virginica
97  virginica
98  virginica
99  virginica

[96 rows x 9 columns]
```

[59]:
```
print(df.replace(6.3,600))
```

```
        a         b         c         d         e         f     g  \
0     5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20
1     4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20
2     4.7  0.111111  3.200000  0.500000  1.300000       NaN  0.20
3     4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20
7     5.0  0.194444  3.400000       NaN  1.500000  0.084746  0.20
..    …       …         …         …         …         …     …
95    7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60
96    7.4       NaN  0.333333  6.100000  0.864407  1.900000  0.75
97    7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00
98    6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20
99  600.0  0.555556  2.800000  0.333333  5.100000  0.694915  1.50
```

```
              h          i
0   0.041666667     setosa
1   0.041666667     setosa
2   0.041666667     setosa
3   0.041666667     setosa
7   0.041666667     setosa
..          …        …
95        0.625  virginica
96    virginica        NaN
97  0.791666667  virginica
98        0.875  virginica
99  0.583333333  virginica

[96 rows x 9 columns]
```

[60]:
```
print(df.replace('.',np.nan))
```

```
      a         b         c         d         e         f     g            h  \
0   5.1  0.222222  3.500000  0.625000  1.400000  0.067797  0.20  0.041666667
1   4.9  0.166667  3.000000  0.416667  1.400000  0.067797  0.20  0.041666667
2   4.7  0.111111  3.200000  0.500000  1.300000       NaN  0.20  0.041666667
3   4.6  0.083333  3.100000  0.458333  1.500000  0.084746  0.20  0.041666667
7   5.0  0.194444  3.400000       NaN  1.500000  0.084746  0.20  0.041666667
..  …       …         …       …         …         …     …           …
95  7.2  0.805556  3.000000  0.416667  5.800000  0.813559  1.60       0.625
96  7.4       NaN  0.333333  6.100000  0.864407  1.900000  0.75   virginica
97  7.9  0.999900  3.800000  0.750000  6.400000  0.915254  2.00  0.791666667
98  6.4  0.583333  2.800000  0.333333  5.600000  0.779661  2.20       0.875
99  6.3  0.555556  2.800000  0.333333  5.100000  0.694915  1.50  0.583333333

           i
0      setosa
1      setosa
2      setosa
3      setosa
7      setosa
..        …
95  virginica
96        NaN
97  virginica
98  virginica
99  virginica

[96 rows x 9 columns]
```

[61]:
```
df[np.random.rand(df.shape[0]>0.5)]=1.5
```

```
␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-61-07651a58e614> in <module>
   ----> 1 df[np.random.rand(df.shape[0]>0.5)]=1.5


       mtrand.pyx in numpy.random.mtrand.RandomState.rand()


       mtrand.pyx in numpy.random.mtrand.RandomState.random_sample()


       _common.pyx in numpy.random._common.double_fill()


       TypeError: an integer is required
```

[62]:
```
#4.2.5 Applying functions

f=lambda df: df.max()-df.min()
```

[64]:
```
print("applying function element wise =\n",df.iloc[: ,3:5].apply(f)) # applying␣
 ↪function element wise
```

```
applying function element wise =
 d    6.090000
e    6.035593
dtype: float64
```

[65]:
```
def sf(x):
    return x.max()-x.min()
```

[66]:
```
print(df.iloc[: ,3:5].apply(sf)) # applying function element wise
```

```
d    6.090000
e    6.035593
dtype: float64
```

[67]:
```
#4.2.6 Group Operations
```

[68]:
```
grouped=df[['a','b','e']].groupby(df['i']) #group according to column 'i'
```

[69]:
```
print(grouped.mean())
```

```
                  a         b         e
i
setosa      5.034483  0.204368  1.471429
versicolor  6.026471  0.479575  4.315152
virginica   6.625000  0.645830  5.634375
```

[70]:
```
grouped=df[['a','b','e']].groupby([df['i'],df['c']]).mean()
```

[71]:
```
print(grouped.unstack())
```

```
               a                                                        \
c            2.9   3.0       3.1       3.2  3.3       3.4       3.5  3.6
i
setosa      4.40  4.75  4.766667  4.700000  5.1  5.085714  5.133333  4.6
versicolor  6.14  6.20  6.800000  6.433333  6.3       NaN       NaN  NaN
virginica   6.80  6.95       NaN  6.750000  6.5       NaN       NaN  7.2

                      …    e
c            3.7  3.8  …  4.0  4.4  2.0   2.2  2.3  2.4  2.5  2.6  2.7   2.8
i                     …
setosa      5.25  5.4  …  1.2  1.5  NaN   NaN  NaN  NaN  NaN  NaN  NaN   NaN
versicolor   NaN  NaN  …  NaN  NaN  3.5  4.25  4.0  3.6  4.4  3.5  4.3  4.52
virginica    NaN  7.8  …  NaN  NaN  NaN  5.00  NaN  NaN  5.1  6.9  5.1  5.40

[3 rows x 63 columns]
```

[72]:
```
#4.2.7 Data Summarizing
```

[73]:
```
print(df['a'].nunique()) # number of distinct values in a column
```

```
33
```

[74]:
```
print(df['a'].value_counts()) # count the number of rows for each unique value
```

```
6.4    6
6.3    6
5.8    5
5.7    5
5.1    5
5.0    4
6.5    4
6.0    4
5.4    4
5.6    4
4.8    4
4.9    4
```

```
6.7    4
5.2    3
5.5    3
7.7    3
6.1    3
7.2    3
6.8    2
4.6    2
6.6    2
5.9    2
4.7    2
6.9    2
6.2    2
7.6    1
7.0    1
4.4    1
7.4    1
7.1    1
7.9    1
4.3    1
7.3    1
Name: a, dtype: int64
```

[75]:
```
print(df.describe()) # descriptive statistics for each column
```

```
                a          b          c          d          e          f  \
count  96.000000  95.000000  96.000000  93.000000  94.000000  95.000000
mean    5.940625   0.451566   2.997222   0.485053   3.880472   0.509489
std     0.856502   0.235253   0.513301   0.616023   1.785482   0.328692
min     4.300000   0.010000   0.333333   0.010000   0.864407   0.010000
25%     5.200000   0.250000   2.800000   0.333333   1.600000   0.110169
50%     5.900000   0.444444   3.000000   0.416667   4.500000   0.593220
75%     6.500000   0.611111   3.300000   0.541667   5.100000   0.694915
max     7.900000   0.999900   4.400000   6.100000   6.900000   1.900000

                g
count  95.000000
mean    1.222632
std     0.743009
min     0.100000
25%     0.400000
50%     1.400000
75%     1.800000
max     2.500000
```

[76]:
```
print(df.mean())
```

```
a    5.940625
```

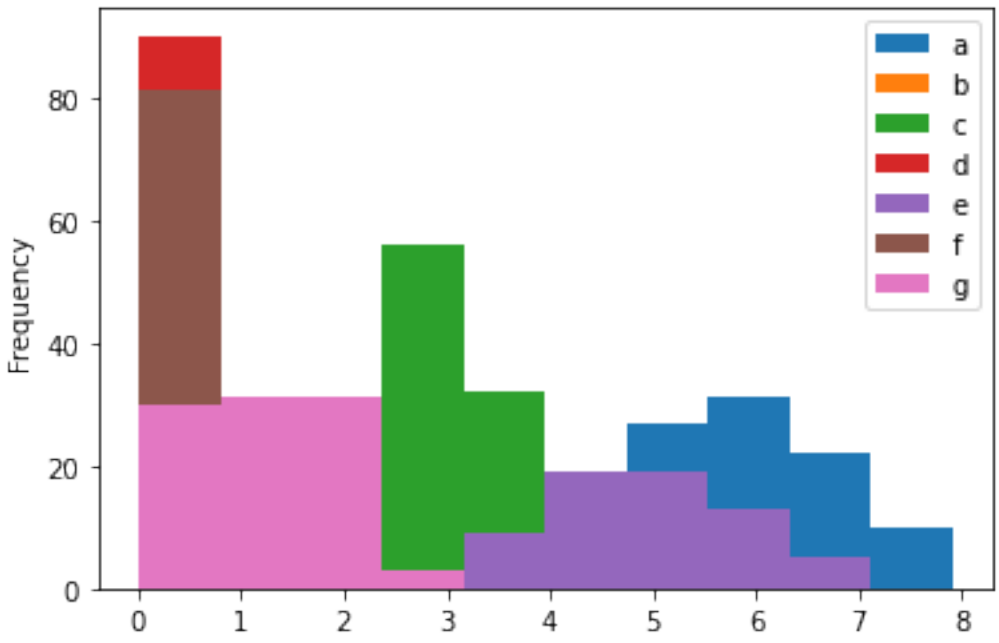
```
b    0.451566
c    2.997222
d    0.485053
e    3.880472
f    0.509489
g    1.222632
dtype: float64
```

[77]:
```
print(df.sort_index().head())
```

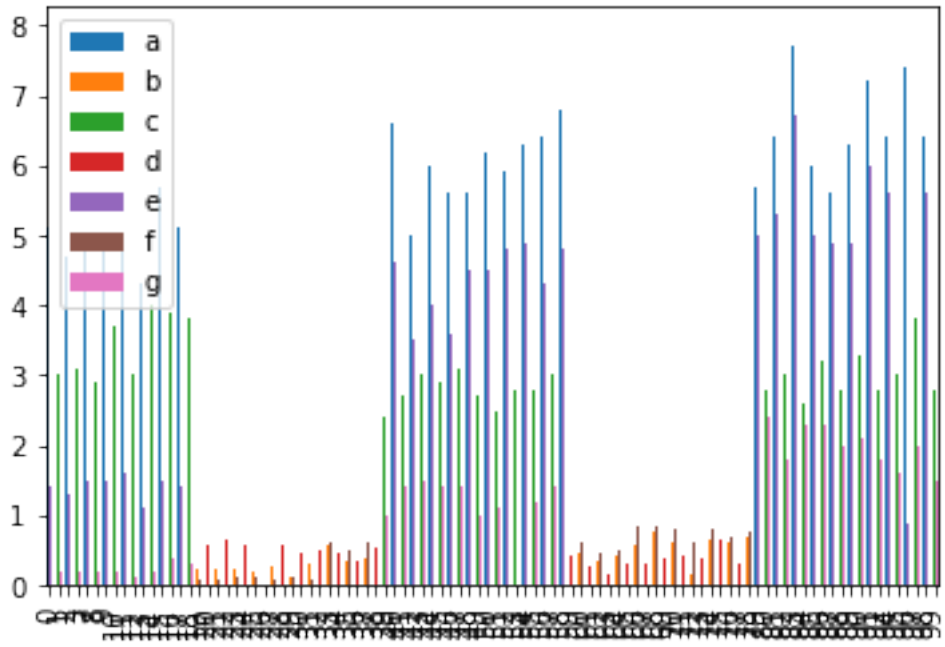
```
     a         b    c         d    e         f    g             h       i
0  5.1  0.222222  3.5  0.625000  1.4  0.067797  0.2  0.041666667  setosa
1  4.9  0.166667  3.0  0.416667  1.4  0.067797  0.2  0.041666667  setosa
2  4.7  0.111111  3.2  0.500000  1.3       NaN  0.2  0.041666667  setosa
3  4.6  0.083333  3.1  0.458333  1.5  0.084746  0.2  0.041666667  setosa
7  5.0  0.194444  3.4       NaN  1.5  0.084746  0.2  0.041666667  setosa
```

[78]:
```
#4.2.8 Data Visualization
```

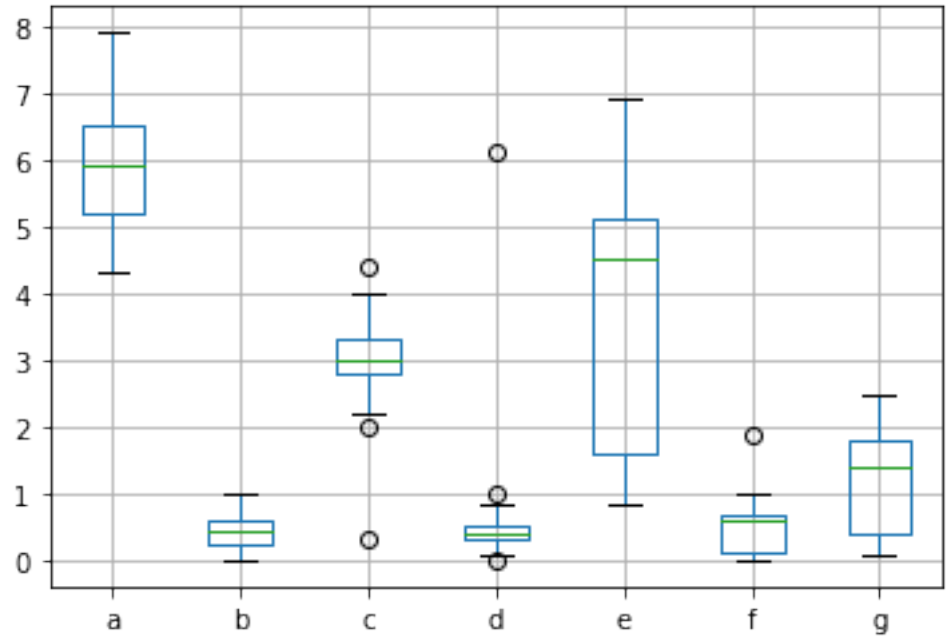
[82]:
```
import matplotlib.pyplot as plt

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



[83]:
```
df.plot(kind='bar')
plt.show()
```



[84]:
```
df.boxplot()
plt.show()
```



[85]:
```
#4.3 Try Out
```

[86]:
```
#1
df=pd.read_csv('E:/University Works/3rd Year/Semester 6/CO 544 - Machine␣
↪Learning and Data Mining/Lab/2/lab02Exercise01.
↪csv',names=['Chanel1','Chanel2','Chanel3','Chanel4','Chanel5'])
```

[87]:
```
print("df = \n",df)
```

```
df =
         Chanel1   Chanel2   Chanel3   Chanel4   Chanel5
0     -0.022098 -0.135461 -0.100475 -0.014574  0.036626
1     -0.021707 -0.164396 -0.106911 -0.027774 -0.045130
2           NaN -0.107590 -0.044757 -0.040040 -0.080305
3      0.014929 -0.016449 -0.001463 -0.045280  0.000612
4     -0.000988  0.005172 -0.052417 -0.054542  0.090948
…           …         …         …         …         …
23993  0.246057  1.039765  1.429124  0.649511  0.953896
23994  0.260120  1.130245  1.510286  0.699971  1.042690
23995  0.286042  1.284954  1.639914  0.766578  1.160491
23996  0.308476  1.392243  1.749650  0.811173  1.232035
23997  0.314683  1.393349  1.792961  0.821080  1.225376

[23998 rows x 5 columns]
```

[88]:
```
#2
```

[89]:
```
print(df.isnull())
```

```
       Chanel1  Chanel2  Chanel3  Chanel4  Chanel5
0        False    False    False    False    False
1        False    False    False    False    False
2         True    False    False    False    False
3        False    False    False    False    False
4        False    False    False    False    False
…          …        …        …        …        …
23993    False    False    False    False    False
23994    False    False    False    False    False
23995    False    False    False    False    False
23996    False    False    False    False    False
23997    False    False    False    False    False

[23998 rows x 5 columns]
```

[90]:
```
print(df.mean())
```

```
Chanel1   -0.000129
Chanel2   -0.000297
Chanel3   -0.000502
Chanel4   -0.000301
```

```
Chanel5   -0.000772
dtype: float64
```

[91]:
```
print(df.fillna(df.mean()))
```

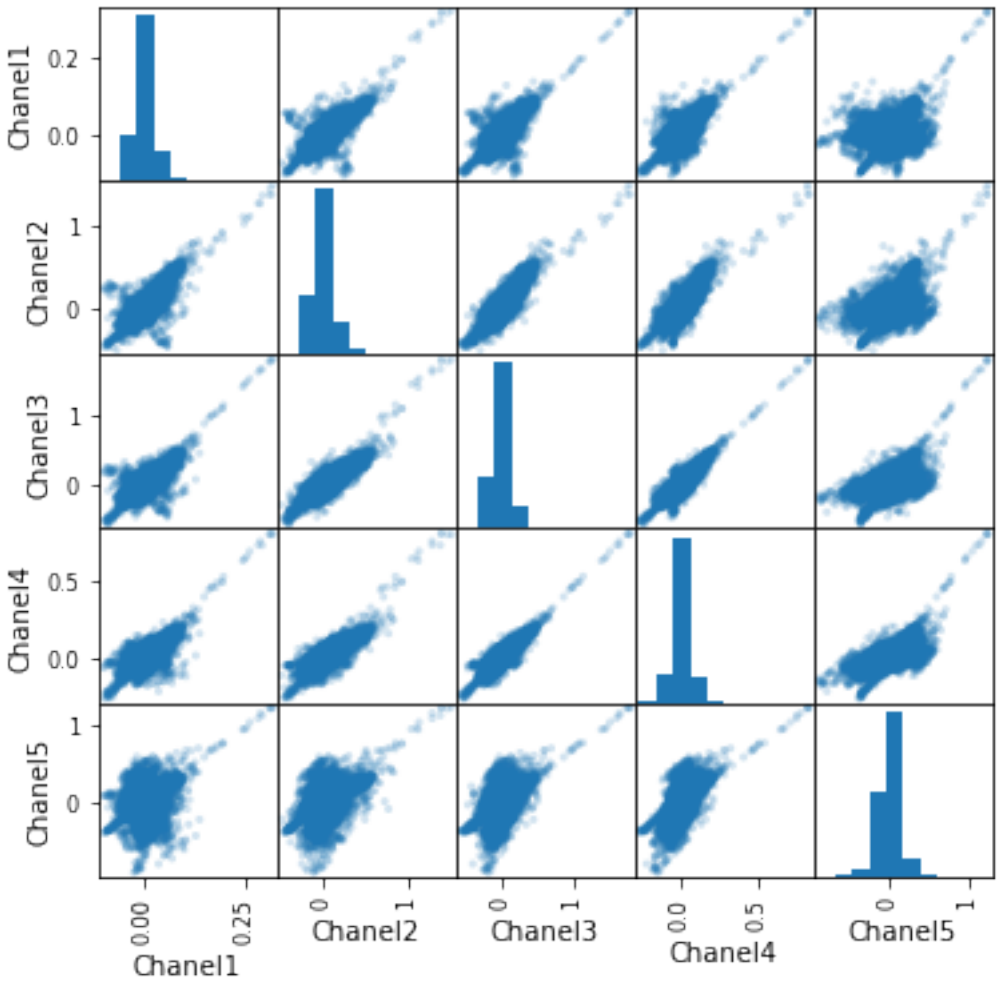
```
        Chanel1   Chanel2   Chanel3   Chanel4   Chanel5
0     -0.022098 -0.135461 -0.100475 -0.014574  0.036626
1     -0.021707 -0.164396 -0.106911 -0.027774 -0.045130
2     -0.000129 -0.107590 -0.044757 -0.040040 -0.080305
3      0.014929 -0.016449 -0.001463 -0.045280  0.000612
4     -0.000988  0.005172 -0.052417 -0.054542  0.090948
…           …         …         …         …         …
23993  0.246057  1.039765  1.429124  0.649511  0.953896
23994  0.260120  1.130245  1.510286  0.699971  1.042690
23995  0.286042  1.284954  1.639914  0.766578  1.160491
23996  0.308476  1.392243  1.749650  0.811173  1.232035
23997  0.314683  1.393349  1.792961  0.821080  1.225376

[23998 rows x 5 columns]
```

[93]:
```
ndf = df.fillna(df.mean())
```

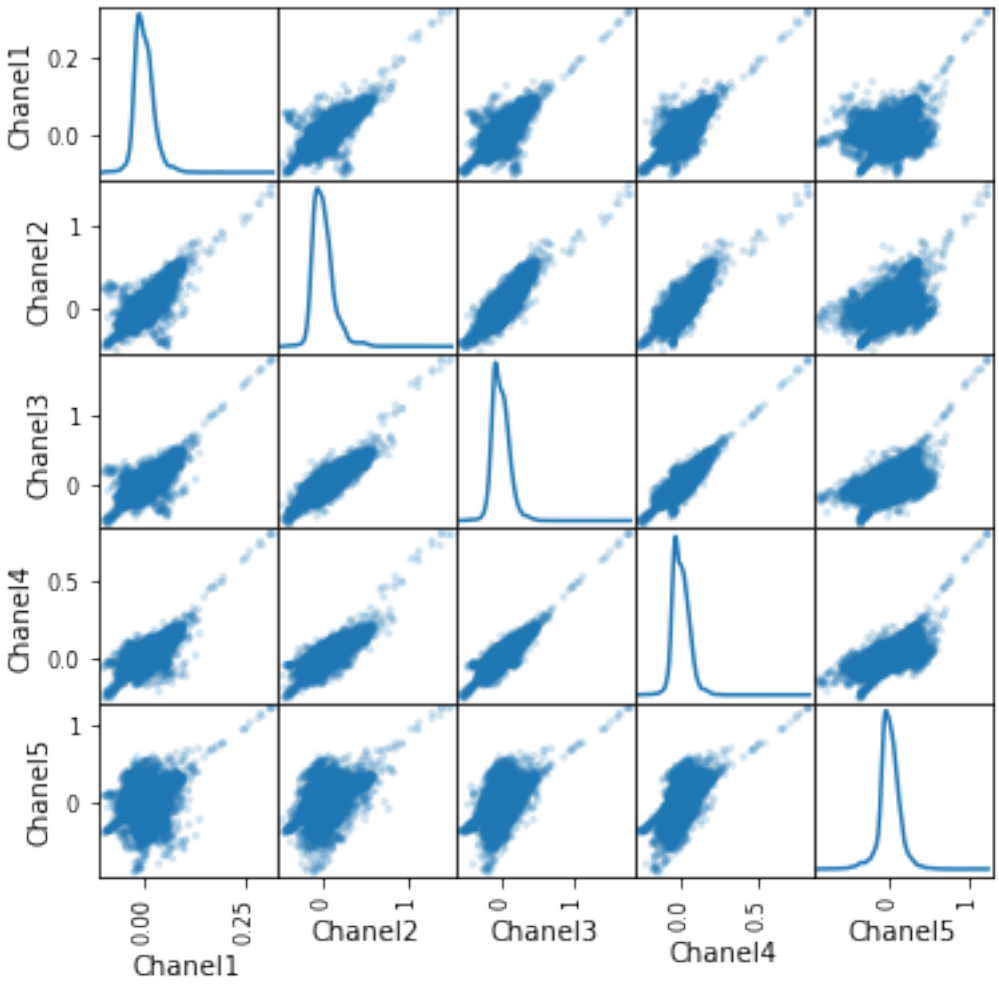
[94]:
```
print("ndf = \n",ndf)
```

```
ndf =
          Chanel1   Chanel2   Chanel3   Chanel4   Chanel5
0     -0.022098 -0.135461 -0.100475 -0.014574  0.036626
1     -0.021707 -0.164396 -0.106911 -0.027774 -0.045130
2     -0.000129 -0.107590 -0.044757 -0.040040 -0.080305
3      0.014929 -0.016449 -0.001463 -0.045280  0.000612
4     -0.000988  0.005172 -0.052417 -0.054542  0.090948
…           …         …         …         …         …
23993  0.246057  1.039765  1.429124  0.649511  0.953896
23994  0.260120  1.130245  1.510286  0.699971  1.042690
23995  0.286042  1.284954  1.639914  0.766578  1.160491
23996  0.308476  1.392243  1.749650  0.811173  1.232035
23997  0.314683  1.393349  1.792961  0.821080  1.225376

[23998 rows x 5 columns]
```

[96]:
```
#3
from pandas.plotting import scatter_matrix
scatter_matrix (ndf , alpha =0.2 , figsize =(6, 6))
plt.show()
```



[97]:
```
from pandas.plotting import scatter_matrix
scatter_matrix (ndf , alpha =0.2 , figsize =(6, 6),diagonal='kde')
plt.show()
```



[98]:
```
#4
```

[99]:
```
#create a new column with some random values
ndf['class']=pd.Series(np.random.randn(ndf.shape[0]),index=ndf.index)
```

[100]:
```
print("ndf = \n",ndf)
```

```
ndf =
          Chanel1   Chanel2   Chanel3   Chanel4   Chanel5     class
0      -0.022098 -0.135461 -0.100475 -0.014574  0.036626  1.118757
1      -0.021707 -0.164396 -0.106911 -0.027774 -0.045130  1.031029
2      -0.000129 -0.107590 -0.044757 -0.040040 -0.080305 -1.267449
3       0.014929 -0.016449 -0.001463 -0.045280  0.000612 -0.616115
4      -0.000988  0.005172 -0.052417 -0.054542  0.090948 -2.470246
...          ...       ...       ...       ...       ...       ...
23993  0.246057  1.039765  1.429124  0.649511  0.953896 -0.565555
```

```
23994  0.260120  1.130245  1.510286  0.699971  1.042690  0.959931
23995  0.286042  1.284954  1.639914  0.766578  1.160491 -0.498741
23996  0.308476  1.392243  1.749650  0.811173  1.232035 -0.645921
23997  0.314683  1.393349  1.792961  0.821080  1.225376  0.805114

[23998 rows x 6 columns]
```

[101]:
```
#logic to fill the new class

newclass = []
i = 0
while i < len(ndf):
    if (ndf.values[i,0]+ndf.values[i,4])/2 < (ndf.values[i,1]+ndf.
 ↪values[i,2]+ndf.values[i,3])/3:
        newclass.append(1)
    else:
        newclass.append(0)
    i = i + 1
```

[102]:
```
ndf['class'] = newclass
```

[103]:
```
#fill the 'class' column with necessary conditions applied
print(ndf)
```

```
        Chanel1   Chanel2   Chanel3   Chanel4   Chanel5  class
0     -0.022098 -0.135461 -0.100475 -0.014574  0.036626      0
1     -0.021707 -0.164396 -0.106911 -0.027774 -0.045130      0
2     -0.000129 -0.107590 -0.044757 -0.040040 -0.080305      0
3      0.014929 -0.016449 -0.001463 -0.045280  0.000612      0
4     -0.000988  0.005172 -0.052417 -0.054542  0.090948      0
…           …         …         …         …         …    …
23993  0.246057  1.039765  1.429124  0.649511  0.953896      1
23994  0.260120  1.130245  1.510286  0.699971  1.042690      1
23995  0.286042  1.284954  1.639914  0.766578  1.160491      1
23996  0.308476  1.392243  1.749650  0.811173  1.232035      1
23997  0.314683  1.393349  1.792961  0.821080  1.225376      1

[23998 rows x 6 columns]
```

[ ]:

[ ]:

[ ]: