# 1 - 2D Matrix

In [ ]:

```
import numpy
import pandas

myarray = numpy.array([[1,2,3],[4,5,6]]) # Creating Array
rownames = ['a','b'] # Naming Row
colnames=['f1','f2','f3'] # Naming Column

mydataframe = pandas.DataFrame(myarray, index = rownames, columns=colnames)
print(mydataframe)
```

```
   f1  f2  f3
a   1   2   3
b   4   5   6
```

In [ ]:

```
import numpy
import pandas

myarray = numpy.array([['a','sandhya',9.6],[4,'shreya',6.5]])
rownames = ['r1','r2']
colnames=['f1','f2','f3']

mydataframe = pandas.DataFrame(myarray, index = rownames, columns=colnames)
print(mydataframe)
```

```
   f1       f2   f3
r1  a  sandhya  9.6
r2  4   shreya  6.5
```

# 2 - Import CSV

In [ ]:

```
# Load csv file using pandas from a specific path or url

from pandas import read_csv

data=read_csv('/content/Diabetes.csv')
colnames=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome']

print(data.shape)
```

```
(768, 9)
```

# 3 - Statistical Summary

In [ ]:

```
# This will give statistics of each column in the dataset.

description = data.describe()

print(description)
```

```
       Pregnancies     Glucose  ...         Age     Outcome
count   768.000000  768.000000  ...  768.000000  768.000000
mean      3.845052  120.894531  ...   33.240885    0.348958
std       3.369578   31.972618  ...   11.760232    0.476951
min       0.000000    0.000000  ...   21.000000    0.000000
25%       1.000000   99.000000  ...   24.000000    0.000000
50%       3.000000  117.000000  ...   29.000000    0.000000
75%       6.000000  140.250000  ...   41.000000    1.000000
max      17.000000  199.000000  ...   81.000000    1.000000

[8 rows x 9 columns]
```

In [ ]:

```
# Size of matrix

print(data.shape)
```

```
(768, 9)
```

In [ ]:

```
# Peek at data

print(data.head(4))
```

```
   Pregnancies  Glucose  BloodPressure  ...  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72  ...                     0.627   50        1
1            1       85             66  ...                     0.351   31        0
2            8      183             64  ...                     0.672   32        1
3            1       89             66  ...                     0.167   21        0
```

```
[4 rows x 9 columns]
```

In [ ]:

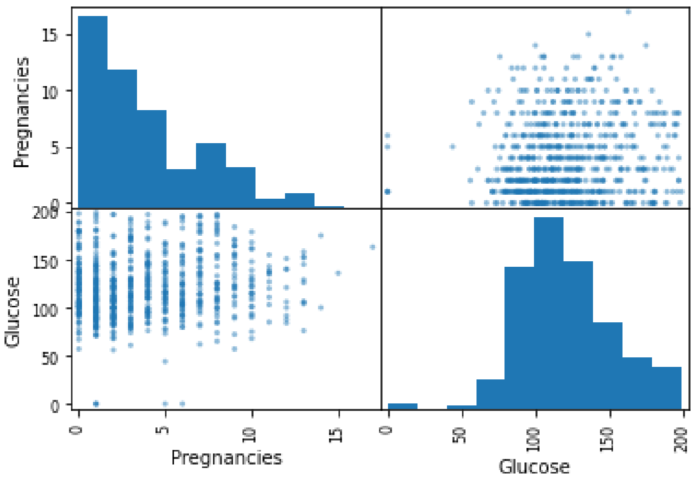
```
# Group on the basis of a particular attribute

print(data.groupby("Outcome").size())
```

```
Outcome
0    500
1    268
dtype: int64
```

## 4 - Data Visualization

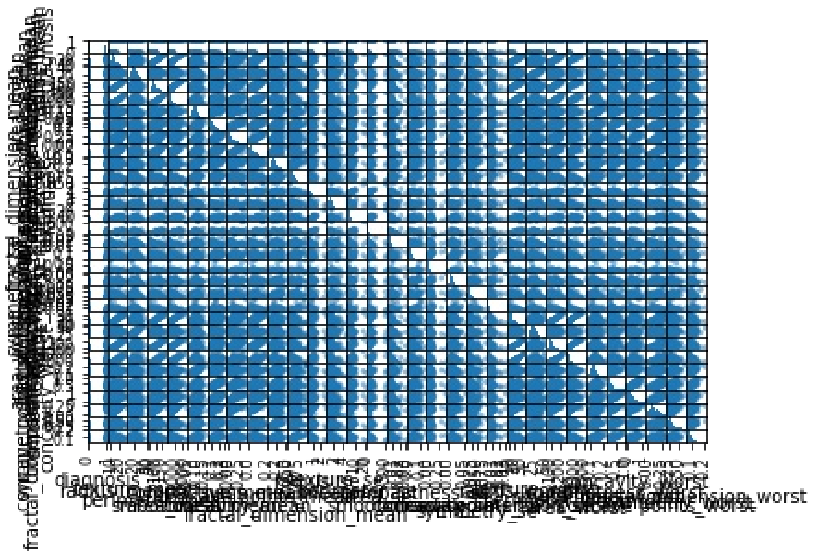
In [ ]:

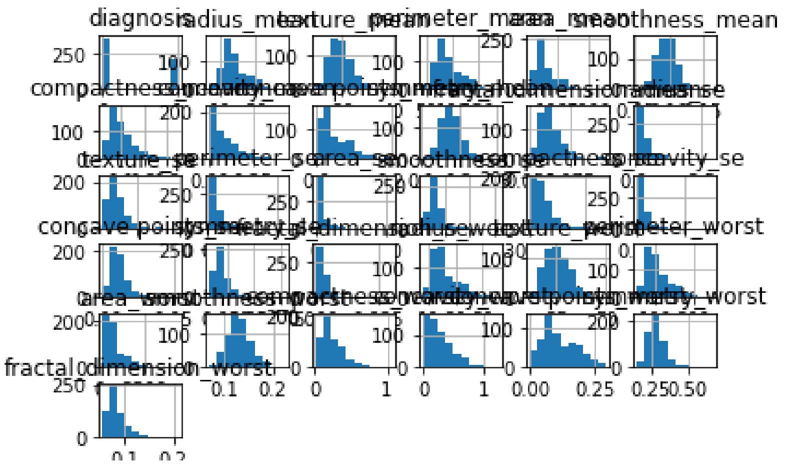
```
# Plotting pairs of attributes as scattered plot, specify the attributes to be plotted explicitly

import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix

scatter_matrix(data[['Pregnancies','Glucose']])
plt.show()
```



In [133]:

```
# Plot all pairs of attributes in data

import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix

scatter_matrix(data) # Scatter plot

print()
plt.show()
print("\n\n")

data.hist() # Histogram
plt.show()
```





## 5 - Standardization of dataset

In [132]:

```
from sklearn.preprocessing import StandardScaler
import pandas
import numpy

arr=data.values  # Convert data frame to array

X=arr[:,0:8] # Split columns
Y=arr[:,8] # Only 7th Column since index starts from 0

scaler=StandardScaler().fit(X) # Fit data for standardization
rescaledX=scaler.transform(X) # Convert the data as per (x-μ)/σ
numpy.set_printoptions(precision=3)

print()
print(rescaledX[0:2,:])
print()

print(X[0:2,:])
```

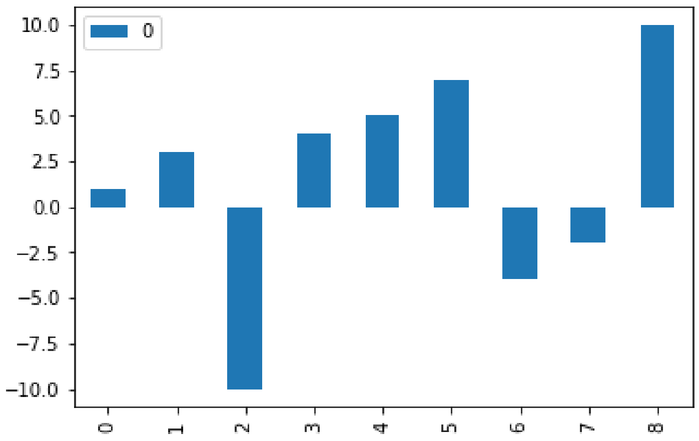
```
[[ 1.298  1.097 -2.073  1.27   0.984  1.568  3.284  2.653]
 [ 1.298  1.83  -0.354  1.686  1.909 -0.827 -0.487 -0.024]]

[[1.000e+00 1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01
  3.001e-01]
 [1.000e+00 2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02
  8.690e-02]]
```

## 6 - Normalizing a Column

In [131]:

```
# Create a dataframe for a set of values in an array

myarray=numpy.array([1,3,-10,4,5,7,-4,-2,10])
mydataframe = pandas.DataFrame(myarray)

print()
print(mydataframe)
print()

# Plot the data

mydataframe.plot(kind='bar')
plt.show()
```

```
    0
0   1
1   3
2 -10
3   4
4   5
5   7
6  -4
7  -2
8  10
```

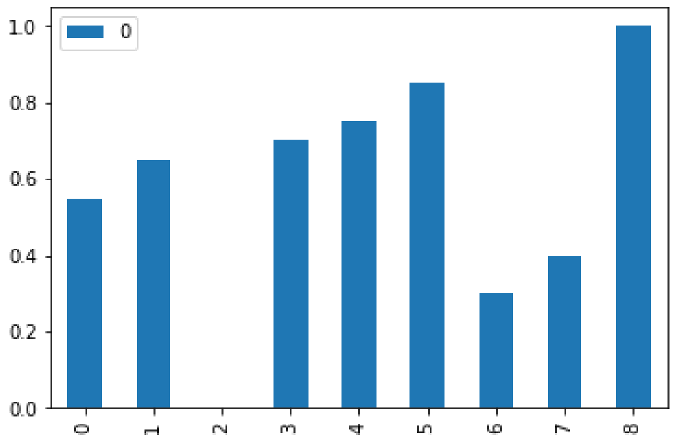


In [134]:

```
# Plot normalized data

from sklearn import preprocessing
fl_x = mydataframe.values.astype(float)

# fl_x=mydataframe[['fl']].values.astype(float)   # If specific feature name is to be converted

min_max_scaler = preprocessing.MinMaxScaler()
X_scaled=min_max_scaler.fit_transform(fl_x)
df_normalized = pandas.DataFrame(X_scaled)

print()
print(df_normalized)
print("\n\n")
```

```
df_normalized.plot(kind='bar')
plt.show()
```

```
      0
0  0.55
1  0.65
2  0.00
3  0.70
4  0.75
5  0.85
6  0.30
7  0.40
8  1.00
```



# Normalization and Standardization

#### Difference between Normalization and Standardization

- **The terms normalization and standardization are sometimes used interchangeably, but they usually refer to different things.**
- **Normalization usually means to scale a variable to have a values between a desired range (like [-1,1] or [0,1]).**
- **Standardization transforms data to have a mean of zero and a standard deviation of 1.**
- **The result of standardization (or Z-score normalization) is that the features will be rescaled to ensure the mean and the standard deviation to be 0 and 1, respectively.**
- **Advantage of Normalization over Standardization is that we are not bound to any specific distribution.**
- **In addition to that Normalization also suppresses the effect of outliers to some extent.**

# Breast Cancer

In [ ]:

```
# Load csv file using pandas from a specific path or url

from pandas import read_csv

data=read_csv('/content/Breast Cancer.csv')

print(data.shape)
```

```
(569, 33)
```

In [ ]:

```
# This will give statistics of each column in the dataset.

description = data.describe()

print(description)
```

```
                 id  radius_mean  ...  fractal_dimension_worst  Unnamed: 32
count  5.690000e+02   569.000000  ...               569.000000          0.0
mean   3.037183e+07    14.127292  ...                 0.083946          NaN
std    1.250206e+08     3.524049  ...                 0.018061          NaN
min    8.670000e+03     6.981000  ...                 0.055040          NaN
25%    8.692180e+05    11.700000  ...                 0.071460          NaN
50%    9.060240e+05    13.370000  ...                 0.080040          NaN
75%    8.813129e+06    15.780000  ...                 0.092080          NaN
max    9.113205e+08    28.110000  ...                 0.207500          NaN

[8 rows x 32 columns]
```

In [ ]:

```
# Size of matrix

print(data.shape)
```

```
(569, 33)
```

In [ ]:

```
# Peek at data

print(data.head(4))
```

```
         id diagnosis  ...  fractal_dimension_worst  Unnamed: 32
```

```
0    842302         M  ...                   0.11890          NaN
1    842517         M  ...                   0.08902          NaN
2  84300903         M  ...                   0.08758          NaN
3  84348301         M  ...                   0.17300          NaN

[4 rows x 33 columns]
```

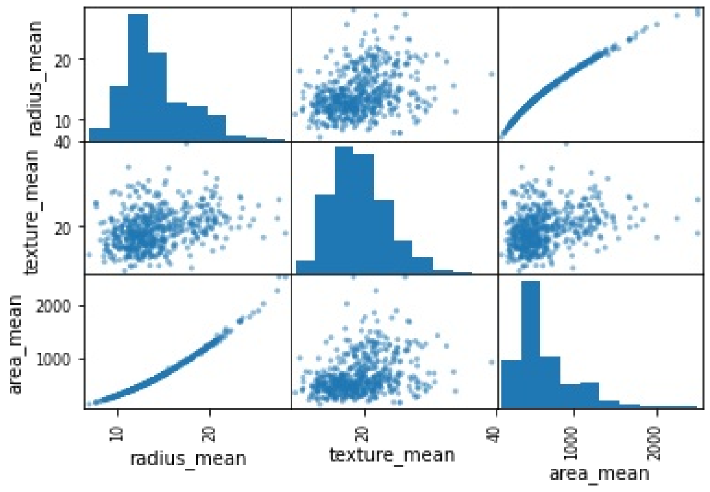
In [ ]:

```
# Group on the basis of a particular attribute

print(data.groupby("perimeter_se").size())
```

```
perimeter_se
0.7570     1
0.7714     1
0.8439     1
0.8484     1
0.8730     1
          ..
10.0500    1
10.1200    1
11.0700    1
18.6500    1
21.9800    1
Length: 533, dtype: int64
```

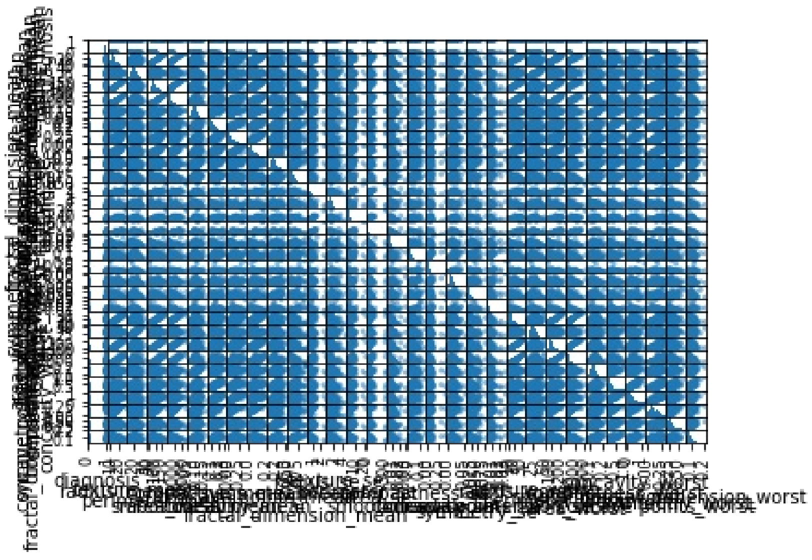
In [ ]:

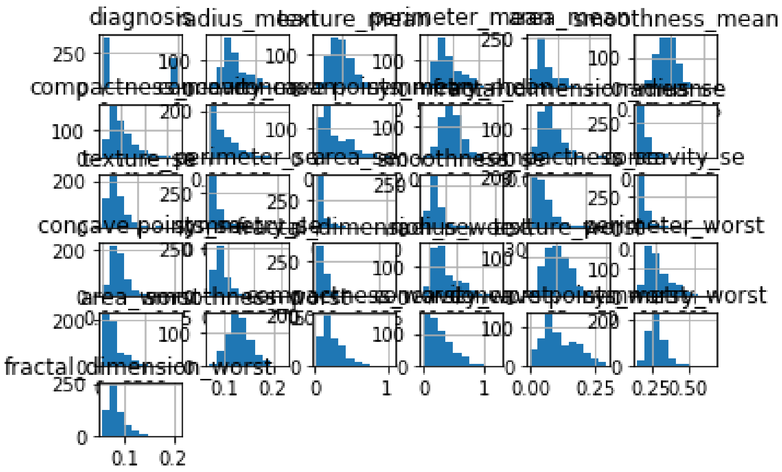
```
# Plotting pairs of attributes as scattered plot, specify the attributes to be plotted explicitly

import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix

scatter_matrix(data[['radius_mean','texture_mean','area_mean']])
plt.show()
```



In [ ]:

```
# Plot all pairs of attributes in data

import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix

# Cleaning and modifying the data
data = data.drop('id',axis=1)
data = data.drop('Unnamed: 32',axis=1)

# Mapping Benign to 0 and Malignant to 1
data['diagnosis'] = data['diagnosis'].map({'M':1,'B':0})

#Check the data stats
data.describe()

scatter_matrix(data)  # Scatter plot
plt.show()

print("\n\n")

data.hist()  # Histogram
plt.show()
```





In [ ]:

```
data.shape
```

Out[ ]:

```
(569, 31)
```

In [128]:

```
from sklearn.preprocessing import StandardScaler
import pandas
import numpy

arr=data.values  # Convert data frame to array

X=arr[:,0:31] # Split columns
Y=arr[:,30]

scaler=StandardScaler().fit(X) # Fit data for standardization
rescaledX=scaler.transform(X) # Convert the data as per (x-μ)/σ
numpy.set_printoptions(precision=3)

print(rescaledX[0:2,:])

print()

print(X[0:2,:])
```

```
[[ 1.298e+00  1.097e+00 -2.073e+00  1.270e+00  9.844e-01  1.568e+00
   3.284e+00  2.653e+00  2.532e+00  2.218e+00  2.256e+00  2.490e+00
  -5.653e-01  2.833e+00  2.488e+00 -2.140e-01  1.317e+00  7.240e-01
   6.608e-01  1.149e+00  9.071e-01  1.887e+00 -1.359e+00  2.304e+00
   2.001e+00  1.308e+00  2.617e+00  2.110e+00  2.296e+00  2.751e+00
   1.937e+00]
 [ 1.298e+00  1.830e+00 -3.536e-01  1.686e+00  1.909e+00 -8.270e-01
  -4.871e-01 -2.385e-02  5.481e-01  1.392e-03 -8.687e-01  4.993e-01
  -8.762e-01  2.633e-01  7.424e-01 -6.054e-01 -6.929e-01 -4.408e-01
   2.602e-01 -8.055e-01 -9.944e-02  1.806e+00 -3.692e-01  1.535e+00
   1.890e+00 -3.756e-01 -4.304e-01 -1.467e-01  1.087e+00 -2.439e-01
   2.812e-01]]

[[1.000e+00 1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01
  3.001e-01 1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00
  1.534e+02 6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03
  2.538e+01 1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01
  2.654e-01 4.601e-01 1.189e-01]
 [1.000e+00 2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02
  8.690e-02 7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00
  7.408e+01 5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03
  2.499e+01 2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01
  1.860e-01 2.750e-01 8.902e-02]]
```