In [1]:

```
import pandas as pd
df = pd.read_csv("C:/Users/hp/Documents/diabetes.csv")
df.head()
```

Out[1]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2

In [2]:

```
"""Traansformation  stages
1 Rescale data
2 Standardize data
3 Normalize data
4 Binarize """
```

Out[2]:

```
'Traansformation  stages\n1 Rescale data\n2 Standardize data\n3 Normalize
data\n4 Binarize '
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [4]:

```
df.describe()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [5]:

```
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.000
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.250
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.000
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.000
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.250
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.600
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.626
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.000
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.000

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insul
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:

```
'''Statiscal challenges with our data
1. Very high scales in the features : Require rescaling
2. Very high standard deviation in the features : requires standardition
3. Non normal distribution : Normalize'''
```

Out[7]:

```
'Statiscal challenges with our data\n1. Very high scales in the features
Require rescaling\n2. Very high standard deviation in the features : requi
res standardition\n3. Non normal distribution : Normalize'
```

In [8]:

```
import matplotlib.pyplot as plt
```

In [9]:

```
# Rescaling
# Rescale data between (0 and 1)
from numpy import set_printoptions
```

In [10]:

```
from numpy import set_printoptions
from sklearn.preprocessing import MinMaxScaler
        #'''We must convert our dataframe into an array before we can implement any fea
ture transformation.
#We call .values on the dataframe
#We do not apply transformations to the target column
#We must seperate the target column from the input columns before transformation'''

     #dfArr = df.values
        # X = dfArr[0:767,0:8]
        # Y = dfArr[0:767,0:8]
 #ax = dfArr [e : 767, 0:8] Y dfArr [0 : 767, 8]
 #X dfArr[: ,0:8] Y dfArr[: ,8]
 #scaler MinMaxScaler (feature_range= (0, 1)) #CaalLing the constructor of the MinMaxSc
aler Class. Specify range
 #rescaledx scaler.fit_transform(X) Sunnarize transformed data set printoptions (precis
ion=1) rescaledX
```

In [11]:

```
dfArr = df.values
dfArr
```

Out[11]:

```
array([[  6.   , 148.   ,  72.   , ...,   0.627,  50.   ,   1.   ],
       [  1.   ,  85.   ,  66.   , ...,   0.351,  31.   ,   0.   ],
       [  8.   , 183.   ,  64.   , ...,   0.672,  32.   ,   1.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,   0.245,  30.   ,   0.   ],
       [  1.   , 126.   ,  60.   , ...,   0.349,  47.   ,   1.   ],
       [  1.   ,  93.   ,  70.   , ...,   0.315,  23.   ,   0.   ]])
```

In [12]:

```
x = dfArr[0:767,0:8] # Input feature
y = dfArr[0:767,8] # Target feature
```

In [13]:

```
x
```

Out[13]:

```
array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
       [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
       [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
       ...,
       [  2.   , 122.   ,  70.   , ...,  36.8  ,   0.34 ,  27.   ],
       [  5.   , 121.   ,  72.   , ...,  26.2  ,   0.245,  30.   ],
       [  1.   , 126.   ,  60.   , ...,  30.1  ,   0.349,  47.   ]])
```

In [14]:

```
y
```

Out[14]:

```
array([1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 1., 1., 1., 1
       1., 0., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 0.,
       0., 0., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 0.,
       0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0.,
       0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1.,
       0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 1., 1., 1., 0., 0.,
       0., 1., 0., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
       0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0.,
       1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1.,
       1., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 0.,
       0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 1., 1.,
       1., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
       1., 0., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0., 1., 1., 0.,
       0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0.,
       0., 0., 1., 1., 1., 0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 0., 0.,
       1., 0., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0., 1., 0., 1., 1.,
       1., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 1., 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 1.,
       1., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0.,
       0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0.,
       1., 0., 0., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 1., 0., 1., 0.,
       1., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 1., 0., 0., 0., 0., 1.,
       1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0.,
       0., 1., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       1., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0.,
       1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1., 1., 0.,
       0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
       1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
       0., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0., 1., 0., 0.,
       1., 0., 0., 1., 0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 1.,
       1., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 0.,
       1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 0., 1., 1.,
       1., 1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1., 0.,
       0., 1., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1.,
       0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 1., 1., 0., 0., 1., 0.,
       0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
       1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0.,
       1., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0., 0., 0.,
       0., 1.])
```

In [15]:

```
# more implicit way, taking all the rows
X= dfArr[:,0:8]
Y = dfArr[:,8]
# Sunnarize transformed data
scaler = MinMaxScaler(feature_range= (0, 1)) #CaalLing the constructor of the MinMaxSca
ler Class. Specify range
rescaledX = scaler.fit_transform(X)
set_printoptions(precision=1) # precision specIFY decimal range
print(rescaledX)
```

```
[[0.4 0.7 0.6 ... 0.5 0.2 0.5]
 [0.1 0.4 0.5 ... 0.4 0.1 0.2]
 [0.5 0.9 0.5 ... 0.3 0.3 0.2]
 ...
 [0.3 0.6 0.6 ... 0.4 0.1 0.2]
 [0.1 0.6 0.5 ... 0.4 0.1 0.4]
 [0.1 0.5 0.6 ... 0.5 0.1 0.  ]]
```

In [16]:

```
df.columns
```

Out[16]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insul
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [17]:

```
# converting the rescale X to dataframe and Adding back Y
rescaledXDF = pd.DataFrame(rescaledX, columns = ['Pregnancies', 'Glucose', 'BloodPressu
re', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
```

In [18]:

```
rescaledXDF.head()
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	

In [19]:

```
# Adding back the outcome  column
print(rescaledXDF)
# add outcome column to rescaledXDF
rescaledXDF["Outcome"] = df["Outcome"]
print(rescaledXDF)
```

```
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       
MI  \
0       0.352941  0.743719       0.590164       0.353535  0.000000  0.5007
45   
1       0.058824  0.427136       0.540984       0.292929  0.000000  0.3964
23   
2       0.470588  0.919598       0.524590       0.000000  0.000000  0.3472
43   
3       0.058824  0.447236       0.540984       0.232323  0.111111  0.4187
78   
4       0.000000  0.688442       0.327869       0.353535  0.198582  0.6423
25   
..           ...       ...            ...            ...       ...       
...   
763     0.588235  0.507538       0.622951       0.484848  0.212766  0.4903
13   
764     0.117647  0.613065       0.573770       0.272727  0.000000  0.5484
35   
765     0.294118  0.608040       0.590164       0.232323  0.132388  0.3904
62   
766     0.058824  0.633166       0.491803       0.000000  0.000000  0.4485
84   
767     0.058824  0.467337       0.573770       0.313131  0.000000  0.4530
55   

     DiabetesPedigreeFunction       Age  
0                    0.234415  0.483333  
1                    0.116567  0.166667  
2                    0.253629  0.183333  
3                    0.038002  0.000000  
4                    0.943638  0.200000  
..                        ...       ...  
763                  0.039710  0.700000  
764                  0.111870  0.100000  
765                  0.071307  0.150000  
766                  0.115713  0.433333  
767                  0.101196  0.033333  

[768 rows x 8 columns]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       B
MI  \
0       0.352941  0.743719       0.590164       0.353535  0.000000  0.5007
45   
1       0.058824  0.427136       0.540984       0.292929  0.000000  0.3964
23   
2       0.470588  0.919598       0.524590       0.000000  0.000000  0.3472
43   
3       0.058824  0.447236       0.540984       0.232323  0.111111  0.4187
78   
4       0.000000  0.688442       0.327869       0.353535  0.198582  0.6423
25   
..           ...       ...            ...            ...       ...       
...   
763     0.588235  0.507538       0.622951       0.484848  0.212766  0.4903
13   
764     0.117647  0.613065       0.573770       0.272727  0.000000  0.5484
35   
765     0.294118  0.608040       0.590164       0.232323  0.132388  0.3904
62   
766     0.058824  0.633166       0.491803       0.000000  0.000000  0.4485
84   
```

```
767      0.058824  0.467337       0.573770       0.313131  0.000000  0.4530
55

     DiabetesPedigreeFunction       Age  Outcome
0                    0.234415  0.483333        1
1                    0.116567  0.166667        0
2                    0.253629  0.183333        1
3                    0.038002  0.000000        0
4                    0.943638  0.200000        1
..                        ...       ...      ...
763                  0.039710  0.700000        0
764                  0.111870  0.100000        0
765                  0.071307  0.150000        0
766                  0.115713  0.433333        1
767                  0.101196  0.033333        0

[768 rows x 9 columns]
```

In [20]:

```
rescaledXDF.describe()
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	0.226180	0.607510	0.566438	0.207439	0.094326	0.476790	
std	0.198210	0.160666	0.158654	0.161134	0.136222	0.117499	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.058824	0.497487	0.508197	0.000000	0.000000	0.406855	
50%	0.176471	0.587940	0.590164	0.232323	0.036052	0.476900	
75%	0.352941	0.704774	0.655738	0.323232	0.150414	0.545455	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

In [21]:

```
rescaledXDF.describe().T
```

Out[21]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	0.226180	0.198210	0.0	0.058824	0.176471	0.352941	1.0
Glucose	768.0	0.607510	0.160666	0.0	0.497487	0.587940	0.704774	1.0
BloodPressure	768.0	0.566438	0.158654	0.0	0.508197	0.590164	0.655738	1.0
SkinThickness	768.0	0.207439	0.161134	0.0	0.000000	0.232323	0.323232	1.0
Insulin	768.0	0.094326	0.136222	0.0	0.000000	0.036052	0.150414	1.0
BMI	768.0	0.476790	0.117499	0.0	0.406855	0.476900	0.545455	1.0
DiabetesPedigreeFunction	768.0	0.168179	0.141473	0.0	0.070773	0.125747	0.234095	1.0
Age	768.0	0.204015	0.196004	0.0	0.050000	0.133333	0.333333	1.0
Outcome	768.0	0.348958	0.476951	0.0	0.000000	0.000000	1.000000	1.0

In [22]:

```
# Standardizing
# Digresion : Demostraiting how Standardizing transform original data [DO NOT DO THIS I
N REAL LIFE]
from sklearn.preprocessing import StandardScaler
from numpy import set_printoptions
dfArr = df.values
X= dfArr[:,0:8]
Y = dfArr[:,8]
scaler = StandardScaler().fit(X)
rescaledX_std = scaler.fit_transform(X)
# Sunnarize transformed data
set_printoptions(precision=3) # precision specIFY decimal range
print(rescaledX_std)
```

```
[[ 0.64   0.848  0.15  ...  0.204  0.468  1.426]
 [-0.845 -1.123 -0.161 ... -0.684 -0.365 -0.191]
 [ 1.234  1.944 -0.264 ... -1.103  0.604 -0.106]
 ...
 [ 0.343  0.003  0.15  ... -0.735 -0.685 -0.276]
 [-0.845  0.16  -0.471 ... -0.24  -0.371  1.171]
 [-0.845 -0.873  0.046 ... -0.202 -0.474 -0.871]]
```

In [23]:

```
# converting the rescale X to dataframe and Adding back Y
rescaledXDF_std = pd.DataFrame(rescaledX_std, columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
rescaledXDF_std.head()
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedig
0	0.639947	0.848324	0.149641	0.907270	-0.692891	0.204013	
1	-0.844885	-1.123396	-0.160546	0.530902	-0.692891	-0.684422	
2	1.233880	1.943724	-0.263941	-1.288212	-0.692891	-1.103255	
3	-0.844885	-0.998208	-0.160546	0.154533	0.123302	-0.494043	
4	-1.141852	0.504055	-1.504687	0.907270	0.765836	1.409746	

In [24]:

```
# Adding back the outcome  column
print(rescaledXDF_std)
# add outcome column to rescaledXDF
rescaledXDF_std["Outcome"] = df["Outcome"]
print(rescaledXDF_std)
```

```
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin
MI  \
0       0.639947  0.848324       0.149641       0.907270 -0.692891  0.2040
13
1      -0.844885 -1.123396      -0.160546       0.530902 -0.692891 -0.6844
22
2       1.233880  1.943724      -0.263941      -1.288212 -0.692891 -1.1032
55
3      -0.844885 -0.998208      -0.160546       0.154533  0.123302 -0.4940
43
4      -1.141852  0.504055      -1.504687       0.907270  0.765836  1.4097
46
..           ...       ...            ...            ...       ...
...
763     1.827813 -0.622642       0.356432       1.722735  0.870031  0.1151
69
764    -0.547919  0.034598       0.046245       0.405445 -0.692891  0.6101
54
765     0.342981  0.003301       0.149641       0.154533  0.279594 -0.7351
90
766    -0.844885  0.159787      -0.470732      -1.288212 -0.692891 -0.2402
05
767    -0.844885 -0.873019       0.046245       0.656358 -0.692891 -0.2021
29

     DiabetesPedigreeFunction       Age
0                    0.468492  1.425995
1                   -0.365061 -0.190672
2                    0.604397 -0.105584
3                   -0.920763 -1.041549
4                    5.484909 -0.020496
..                        ...       ...
763                 -0.908682  2.532136
764                 -0.398282 -0.531023
765                 -0.685193 -0.275760
766                 -0.371101  1.170732
767                 -0.473785 -0.871374

[768 rows x 8 columns]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin        B
MI  \
0       0.639947  0.848324       0.149641       0.907270 -0.692891  0.2040
13
1      -0.844885 -1.123396      -0.160546       0.530902 -0.692891 -0.6844
22
2       1.233880  1.943724      -0.263941      -1.288212 -0.692891 -1.1032
55
3      -0.844885 -0.998208      -0.160546       0.154533  0.123302 -0.4940
43
4      -1.141852  0.504055      -1.504687       0.907270  0.765836  1.4097
46
..           ...       ...            ...            ...       ...
...
763     1.827813 -0.622642       0.356432       1.722735  0.870031  0.1151
69
764    -0.547919  0.034598       0.046245       0.405445 -0.692891  0.6101
54
765     0.342981  0.003301       0.149641       0.154533  0.279594 -0.7351
90
766    -0.844885  0.159787      -0.470732      -1.288212 -0.692891 -0.2402
05
```

```
767    -0.844885 -0.873019       0.046245       0.656358 -0.692891 -0.2021
29

     DiabetesPedigreeFunction       Age  Outcome
0                    0.468492  1.425995        1
1                   -0.365061 -0.190672        0
2                    0.604397 -0.105584        1
3                   -0.920763 -1.041549        0
4                    5.484909 -0.020496        1
..                        ...       ...      ...
763                 -0.908682  2.532136        0
764                 -0.398282 -0.531023        0
765                 -0.685193 -0.275760        0
766                 -0.371101  1.170732        1
767                 -0.473785 -0.871374        0

[768 rows x 9 columns]
```

In [25]:

```
rescaledXDF_std.describe()
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	I
count	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e+02	7.680000e
mean	2.544261e-17	3.614007e-18	-1.327244e-17	7.994184e-17	-3.556183e-17	2.295979e
std	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e+00	1.000652e
min	-1.141852e+00	-3.783654e+00	-3.572597e+00	-1.288212e+00	-6.928906e-01	-4.060474e
25%	-8.448851e-01	-6.852363e-01	-3.673367e-01	-1.288212e+00	-6.928906e-01	-5.955785e
50%	-2.509521e-01	-1.218877e-01	1.496408e-01	1.545332e-01	-4.280622e-01	9.419788e
75%	6.399473e-01	6.057709e-01	5.632228e-01	7.190857e-01	4.120079e-01	5.847705e
max	3.906578e+00	2.444478e+00	2.734528e+00	4.921866e+00	6.652839e+00	4.455807e

In [26]:

```
rescaledXDF_std.describe().T
```

Out[26]:

	count	mean	std	min	25%	50%	7
Pregnancies	768.0	2.544261e-17	1.000652	-1.141852	-0.844885	-0.250952	0.639
Glucose	768.0	3.614007e-18	1.000652	-3.783654	-0.685236	-0.121888	0.605
BloodPressure	768.0	-1.327244e-17	1.000652	-3.572597	-0.367337	0.149641	0.563
SkinThickness	768.0	7.994184e-17	1.000652	-1.288212	-1.288212	0.154533	0.719
Insulin	768.0	-3.556183e-17	1.000652	-0.692891	-0.692891	-0.428062	0.412
BMI	768.0	2.295979e-16	1.000652	-4.060474	-0.595578	0.000942	0.584
DiabetesPedigreeFunction	768.0	2.398978e-16	1.000652	-1.189553	-0.688969	-0.300128	0.466
Age	768.0	1.857600e-16	1.000652	-1.041549	-0.786286	-0.360847	0.660
Outcome	768.0	3.489583e-01	0.476951	0.000000	0.000000	0.000000	1.000

In [27]:

```
# compare the standardevation from the original data
df.describe().T
```

Out[27]:

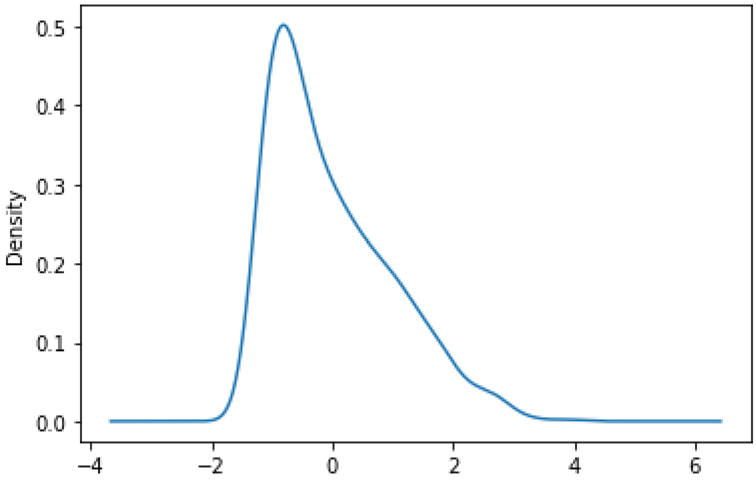
	count	mean	std	min	25%	50%	75
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.000
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.250
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.000
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.000
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.250
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.600
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.626
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.000
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.000

In [28]:

```
rescaledXDF_std.Pregnancies.plot(kind="density")
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1de76b6f7c0>
```



In [29]:

```
# STANDARDIZING THE RESCALED DATA
# CONTINUING TRANSFORMATION USING THE OUTPUT THE RESCALING TRANSFORMATIONSTAGE
# The data used for transformstion rescale stage is use for standardizatin
rescaledXDF
```

Out[29]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedig
0	0.352941	0.743719	0.590164	0.353535	0.000000	0.500745	
1	0.058824	0.427136	0.540984	0.292929	0.000000	0.396423	
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.347243	
3	0.058824	0.447236	0.540984	0.232323	0.111111	0.418778	
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	
...	...	...	...	...	...	...	
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	
767	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	

768 rows × 9 columns

In [30]:

```
# STANDARDIZATION
from sklearn.preprocessing import StandardScaler
from numpy import set_printoptions

rescaledXDF_Arr = rescaledXDF.values
X = rescaledXDF_Arr[:,0:8]
Y = rescaledXDF_Arr[:,8]
scaler = StandardScaler().fit(X)
rescaledX_std = scaler.fit_transform(X)
# Sunnarize transformed data
set_printoptions(precision=3) # precision specIFY decimal range
print(rescaledX_std)

# converting the rescale X to dataframe and Adding back Y
rescaledXDF_R_S = pd.DataFrame(rescaledX_std, columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
# Adding back the outcome  column
print(rescaledXDF_R_S)
# add outcome column to rescaledXDF
rescaledXDF_R_S["Outcome"] = rescaledXDF["Outcome"]
print(rescaledXDF_R_S)
```

```
[[ 0.64   0.848  0.15  ...  0.204  0.468  1.426]
 [-0.845 -1.123 -0.161 ... -0.684 -0.365 -0.191]
 [ 1.234  1.944 -0.264 ... -1.103  0.604 -0.106]
 ...
 [ 0.343  0.003  0.15  ... -0.735 -0.685 -0.276]
 [-0.845  0.16  -0.471 ... -0.24  -0.371  1.171]
 [-0.845 -0.873  0.046 ... -0.202 -0.474 -0.871]]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       B
MI  \
0       0.639947  0.848324       0.149641       0.907270 -0.692891  0.2040
13
1      -0.844885 -1.123396      -0.160546       0.530902 -0.692891 -0.6844
22
2       1.233880  1.943724      -0.263941      -1.288212 -0.692891 -1.1032
55
3      -0.844885 -0.998208      -0.160546       0.154533  0.123302 -0.4940
43
4      -1.141852  0.504055      -1.504687       0.907270  0.765836  1.4097
46
..           ...       ...            ...            ...       ...
...
763     1.827813 -0.622642       0.356432       1.722735  0.870031  0.1151
69
764    -0.547919  0.034598       0.046245       0.405445 -0.692891  0.6101
54
765     0.342981  0.003301       0.149641       0.154533  0.279594 -0.7351
90
766    -0.844885  0.159787      -0.470732      -1.288212 -0.692891 -0.2402
05
767    -0.844885 -0.873019       0.046245       0.656358 -0.692891 -0.2021
29

     DiabetesPedigreeFunction       Age
0                    0.468492  1.425995
1                   -0.365061 -0.190672
2                    0.604397 -0.105584
3                   -0.920763 -1.041549
4                    5.484909 -0.020496
..                        ...       ...
763                 -0.908682  2.532136
764                 -0.398282 -0.531023
765                 -0.685193 -0.275760
766                 -0.371101  1.170732
767                 -0.473785 -0.871374

[768 rows x 8 columns]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       B
MI  \
0       0.639947  0.848324       0.149641       0.907270 -0.692891  0.2040
13
1      -0.844885 -1.123396      -0.160546       0.530902 -0.692891 -0.6844
22
2       1.233880  1.943724      -0.263941      -1.288212 -0.692891 -1.1032
55
3      -0.844885 -0.998208      -0.160546       0.154533  0.123302 -0.4940
43
4      -1.141852  0.504055      -1.504687       0.907270  0.765836  1.4097
46
..           ...       ...            ...            ...       ...
...
763     1.827813 -0.622642       0.356432       1.722735  0.870031  0.1151
```

```
69
764    -0.547919  0.034598       0.046245       0.405445 -0.692891  0.6101
54
765     0.342981  0.003301       0.149641       0.154533  0.279594 -0.7351
90
766    -0.844885  0.159787      -0.470732      -1.288212 -0.692891 -0.2402
05
767    -0.844885 -0.873019       0.046245       0.656358 -0.692891 -0.2021
29

     DiabetesPedigreeFunction       Age  Outcome
0                    0.468492  1.425995        1
1                   -0.365061 -0.190672        0
2                    0.604397 -0.105584        1
3                   -0.920763 -1.041549        0
4                    5.484909 -0.020496        1
..                        ...       ...      ...
763                 -0.908682  2.532136        0
764                 -0.398282 -0.531023        0
765                 -0.685193 -0.275760        0
766                 -0.371101  1.170732        1
767                 -0.473785 -0.871374        0

[768 rows x 9 columns]
```

In [31]:

```
# INSPECTING THE RESCALED AND STANDRDIZED DATA
rescaledXDF_R_S.describe().T
```

Out[31]:

	count	mean	std	min	25%	50%	7
Pregnancies	768.0	5.493291e-17	1.000652	-1.141852	-0.844885	-0.250952	0.639
Glucose	768.0	2.620878e-16	1.000652	-3.783654	-0.685236	-0.121888	0.605
BloodPressure	768.0	4.771845e-16	1.000652	-3.572597	-0.367337	0.149641	0.563
SkinThickness	768.0	-1.750625e-16	1.000652	-1.288212	-1.288212	0.154533	0.719
Insulin	768.0	9.569891e-17	1.000652	-0.692891	-0.692891	-0.428062	0.412
BMI	768.0	-2.201653e-16	1.000652	-4.060474	-0.595578	0.000942	0.584
DiabetesPedigreeFunction	768.0	3.866988e-17	1.000652	-1.189553	-0.688969	-0.300128	0.466
Age	768.0	4.770490e-18	1.000652	-1.041549	-0.786286	-0.360847	0.660
Outcome	768.0	3.489583e-01	0.476951	0.000000	0.000000	0.000000	1.000

In [32]:

```
# compare to rescale rescaledXDF and rescaledXDF_R_S
rescaledXDF.describe().T
```

Out[32]:

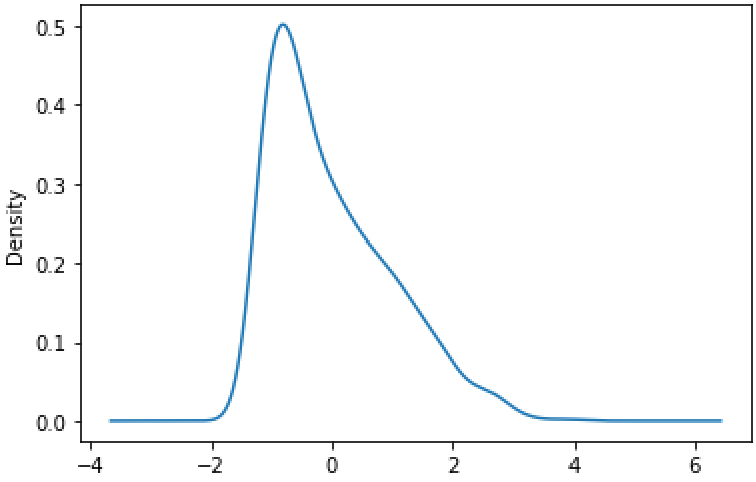
	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	0.226180	0.198210	0.0	0.058824	0.176471	0.352941	1.0
Glucose	768.0	0.607510	0.160666	0.0	0.497487	0.587940	0.704774	1.0
BloodPressure	768.0	0.566438	0.158654	0.0	0.508197	0.590164	0.655738	1.0
SkinThickness	768.0	0.207439	0.161134	0.0	0.000000	0.232323	0.323232	1.0
Insulin	768.0	0.094326	0.136222	0.0	0.000000	0.036052	0.150414	1.0
BMI	768.0	0.476790	0.117499	0.0	0.406855	0.476900	0.545455	1.0
DiabetesPedigreeFunction	768.0	0.168179	0.141473	0.0	0.070773	0.125747	0.234095	1.0
Age	768.0	0.204015	0.196004	0.0	0.050000	0.133333	0.333333	1.0
Outcome	768.0	0.348958	0.476951	0.0	0.000000	0.000000	1.000000	1.0

In [33]:

```
# Not still normal
rescaledXDF_R_S.Pregnancies.plot(kind="density")
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1de73e98f70>
```



In [34]:

```
# IMPLEMENTING NORMALIZATION  ON THE RESCALED  AND STANDARDIZED DATA
from sklearn.preprocessing  import Normalizer
from numpy import set_printoptions
rescaledXDF_R_S_Arr = rescaledXDF_R_S.values
X= rescaledXDF_R_S_Arr[:,0:8]
Y = rescaledXDF_R_S_Arr[:,8]
scaler = Normalizer().fit(X)
NormalizedDX = scaler.fit_transform(X)
# Sunnarize transformed data
set_printoptions(precision=3) # precision specIFY decimal range
print(NormalizedDX)

# converting the rescale X to dataframe and Adding back Y
normalizedX_R_S_N = pd.DataFrame(NormalizedDX, columns = ['Pregnancies', 'Glucose', 'Bl
oodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
# Adding back the outcome  column
print(normalizedX_R_S_N)
# add outcome column to rescaledXDF
normalizedX_R_S_N["Outcome"] = rescaledXDF_R_S["Outcome"]
print(normalizedX_R_S_N)
```

```
[[ 0.294  0.389  0.069 ...  0.094  0.215  0.654]
 [-0.458 -0.609 -0.087 ... -0.371 -0.198 -0.103]
 [ 0.409  0.644 -0.087 ... -0.366  0.2   -0.035]
 ...
 [ 0.298  0.003  0.13  ... -0.638 -0.595 -0.239]
 [-0.391  0.074 -0.218 ... -0.111 -0.172  0.542]
 [-0.457 -0.473  0.025 ... -0.109 -0.256 -0.472]]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       B
MI  \
0       0.293647  0.389263       0.068664       0.416311 -0.317941  0.0936
14
1      -0.458093 -0.609101      -0.087047       0.287852 -0.375682 -0.3710
91
2       0.408951  0.644218      -0.087479      -0.426959 -0.229648 -0.3656
57
3      -0.425010 -0.502137      -0.080761       0.077736  0.062026 -0.2485
23
4      -0.186954  0.082528      -0.246360       0.148546  0.125389  0.2308
16
..           ...       ...            ...            ...       ...
...
763     0.474619 -0.161678       0.092553       0.447334  0.225917  0.0299
05
764    -0.412899  0.026072       0.034849       0.305535 -0.522147  0.4597
99
765     0.297611  0.002864       0.129846       0.134092  0.242609 -0.6379
39
766    -0.391110  0.073968      -0.217909      -0.596333 -0.320749 -0.1111
94
767    -0.457286 -0.472513       0.025030       0.355247 -0.375020 -0.1094
00

     DiabetesPedigreeFunction       Age
0                    0.214973  0.654334
1                   -0.197934 -0.103381
2                    0.200318 -0.034994
3                   -0.463179 -0.523940
4                    0.898036 -0.003356
..                        ...       ...
763                 -0.235953  0.657508
764                 -0.300137 -0.400167
765                 -0.594556 -0.239282
766                 -0.171788  0.541949
767                 -0.256432 -0.471623

[768 rows x 8 columns]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       B
MI  \
0       0.293647  0.389263       0.068664       0.416311 -0.317941  0.0936
14
1      -0.458093 -0.609101      -0.087047       0.287852 -0.375682 -0.3710
91
2       0.408951  0.644218      -0.087479      -0.426959 -0.229648 -0.3656
57
3      -0.425010 -0.502137      -0.080761       0.077736  0.062026 -0.2485
23
4      -0.186954  0.082528      -0.246360       0.148546  0.125389  0.2308
16
..           ...       ...            ...            ...       ...
...
763     0.474619 -0.161678       0.092553       0.447334  0.225917  0.0299
```

```
05
764    -0.412899  0.026072       0.034849       0.305535 -0.522147  0.4597
99
765     0.297611  0.002864       0.129846       0.134092  0.242609 -0.6379
39
766    -0.391110  0.073968      -0.217909      -0.596333 -0.320749 -0.1111
94
767    -0.457286 -0.472513       0.025030       0.355247 -0.375020 -0.1094
00

     DiabetesPedigreeFunction       Age  Outcome
0                    0.214973  0.654334        1
1                   -0.197934 -0.103381        0
2                    0.200318 -0.034994        1
3                   -0.463179 -0.523940        0
4                    0.898036 -0.003356        1
..                        ...       ...      ...
763                 -0.235953  0.657508        0
764                 -0.300137 -0.400167        0
765                 -0.594556 -0.239282        0
766                 -0.171788  0.541949        1
767                 -0.256432 -0.471623        0

[768 rows x 9 columns]
```

In [35]:

```
normalizedX_R_S_N.describe()
```

Out[35]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	-0.029202	-0.035199	0.015273	0.008376	-0.020091	-0.013928	
std	0.372804	0.370004	0.304359	0.386974	0.328127	0.338886	
min	-0.778874	-0.917873	-0.891431	-0.829106	-0.644321	-0.909906	
25%	-0.329314	-0.302614	-0.137178	-0.324046	-0.272916	-0.270521	
50%	-0.116355	-0.051836	0.049074	0.057186	-0.147720	0.000576	
75%	0.266900	0.226748	0.215058	0.327209	0.187654	0.222935	
max	0.891015	0.936682	0.772424	0.818977	0.899737	0.851350	

In [36]:

```
normalizedX_R_S_N.describe().T
```

Out[36]:

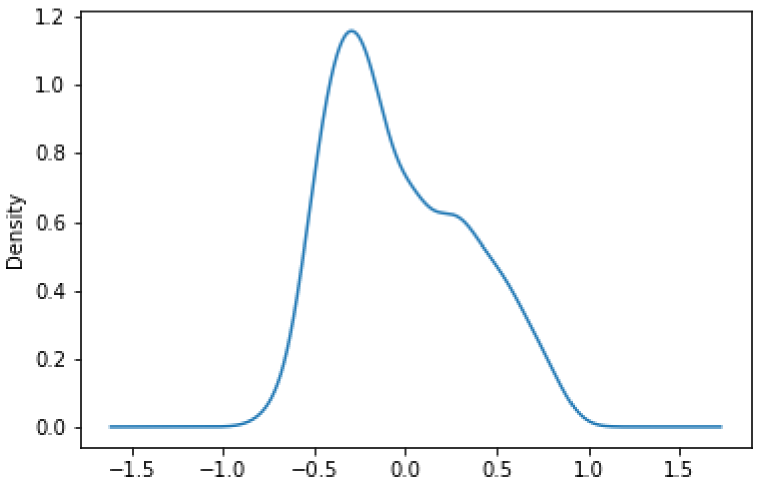
	count	mean	std	min	25%	50%	75%
Pregnancies	768.0	-0.029202	0.372804	-0.778874	-0.329314	-0.116355	0.26690
Glucose	768.0	-0.035199	0.370004	-0.917873	-0.302614	-0.051836	0.22674
BloodPressure	768.0	0.015273	0.304359	-0.891431	-0.137178	0.049074	0.21505
SkinThickness	768.0	0.008376	0.386974	-0.829106	-0.324046	0.057186	0.32720
Insulin	768.0	-0.020091	0.328127	-0.644321	-0.272916	-0.147720	0.18765
BMI	768.0	-0.013928	0.338886	-0.909906	-0.270521	0.000576	0.22293
DiabetesPedigreeFunction	768.0	-0.033374	0.342877	-0.773632	-0.290049	-0.123026	0.18624
Age	768.0	-0.049970	0.369252	-0.732146	-0.355483	-0.140985	0.22795
Outcome	768.0	0.348958	0.476951	0.000000	0.000000	0.000000	1.00000

In [37]:

```
# testing if everything is normal destributed
normalizedX_R_S_N.Pregnancies.plot(kind="density")
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1de77336a00>
```



In [38]:

```
# DIGRESSIONS
# TESTING THE  EFFECT OF ONLY NORMALIZATION ON THE STASTICAL BEHAVIOUR OF THE ORIGINAL DATAFRAME
# IMPLEMENTING NORMALIZATION ON RESCALED AND STANDARDIZED DATA
from sklearn.preprocessing  import Normalizer
from numpy import set_printoptions
rescaledXDF = df.values
X= rescaledXDF[:,0:8]
Y = rescaledXDF[:,8]
scaler = Normalizer().fit(X)
NormalizedX_N = scaler.fit_transform(X)
# Sunnarize transformed data
set_printoptions(precision=3) # precision specIFY decimal range
print(NormalizedX_N)

# converting the rescale X to dataframe and Adding back Y
NormalizedX_N = pd.DataFrame(NormalizedX_N, columns = ['Pregnancies', 'Glucose', 'Blood Pressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
# Adding back the outcome  column
print(NormalizedX_N)
# add outcome column to rescaledXDF
NormalizedX_N["Outcome"] = df["Outcome"]
print(NormalizedX_N)
```

```
[[0.034 0.828 0.403 ... 0.188 0.004 0.28 ]
 [0.008 0.716 0.556 ... 0.224 0.003 0.261]
 [0.04  0.924 0.323 ... 0.118 0.003 0.162]
 ...
 [0.027 0.651 0.388 ... 0.141 0.001 0.161]
 [0.007 0.838 0.399 ... 0.2   0.002 0.313]
 [0.008 0.736 0.554 ... 0.241 0.002 0.182]]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       B
MI  \
0       0.033552  0.827625       0.402628       0.195722  0.000000  0.1878
93
1       0.008424  0.716040       0.555984       0.244296  0.000000  0.2240
79
2       0.040398  0.924097       0.323181       0.000000  0.000000  0.1176
58
3       0.006612  0.588467       0.436392       0.152076  0.621527  0.1857
97
4       0.000000  0.596386       0.174127       0.152361  0.731335  0.1876
22
..           ...       ...            ...            ...       ...     
...
763     0.042321  0.427443       0.321640       0.203141  0.761779  0.1392
36
764     0.013304  0.811526       0.465629       0.179600  0.000000  0.2447
88
765     0.026915  0.651352       0.387582       0.123811  0.602905  0.1410
37
766     0.006653  0.838285       0.399184       0.000000  0.000000  0.2002
57
767     0.007915  0.736052       0.554018       0.245351  0.000000  0.2406
02

     DiabetesPedigreeFunction       Age
0                    0.003506  0.279603
1                    0.002957  0.261144
2                    0.003393  0.161591
3                    0.001104  0.138852
4                    0.009960  0.143655
..                        ...       ...
763                  0.000724  0.266623
764                  0.002262  0.179600
765                  0.001319  0.161492
766                  0.002322  0.312694
767                  0.002493  0.182034

[768 rows x 8 columns]
     Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       B
MI  \
0       0.033552  0.827625       0.402628       0.195722  0.000000  0.1878
93
1       0.008424  0.716040       0.555984       0.244296  0.000000  0.2240
79
2       0.040398  0.924097       0.323181       0.000000  0.000000  0.1176
58
3       0.006612  0.588467       0.436392       0.152076  0.621527  0.1857
97
4       0.000000  0.596386       0.174127       0.152361  0.731335  0.1876
22
..           ...       ...            ...            ...       ...     
...
763     0.042321  0.427443       0.321640       0.203141  0.761779  0.1392
```

```
36
764     0.013304  0.811526       0.465629       0.179600  0.000000  0.2447
88
765     0.026915  0.651352       0.387582       0.123811  0.602905  0.1410
37
766     0.006653  0.838285       0.399184       0.000000  0.000000  0.2002
57
767     0.007915  0.736052       0.554018       0.245351  0.000000  0.2406
02

     DiabetesPedigreeFunction       Age  Outcome
0                    0.003506  0.279603        1
1                    0.002957  0.261144        0
2                    0.003393  0.161591        1
3                    0.001104  0.138852        0
4                    0.009960  0.143655        1
..                        ...       ...      ...
763                  0.000724  0.266623        0
764                  0.002262  0.179600        0
765                  0.001319  0.161492        0
766                  0.002322  0.312694        1
767                  0.002493  0.182034        0

[768 rows x 9 columns]
```

In [39]:

```
NormalizedX_N.describe()
```

Out[39]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	0.022645	0.682903	0.402801	0.112086	0.318921	0.186874	
std	0.020956	0.161166	0.153428	0.092546	0.338570	0.063402	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.006476	0.587637	0.317522	0.000000	0.000000	0.146291	
50%	0.016716	0.704501	0.430685	0.114464	0.249215	0.186167	
75%	0.033330	0.801606	0.511070	0.181524	0.632833	0.226831	
max	0.117208	0.973682	0.848036	0.419691	0.970458	0.400734	

In [40]:

```
NormalizedX_N.describe().T
```

Out[40]:

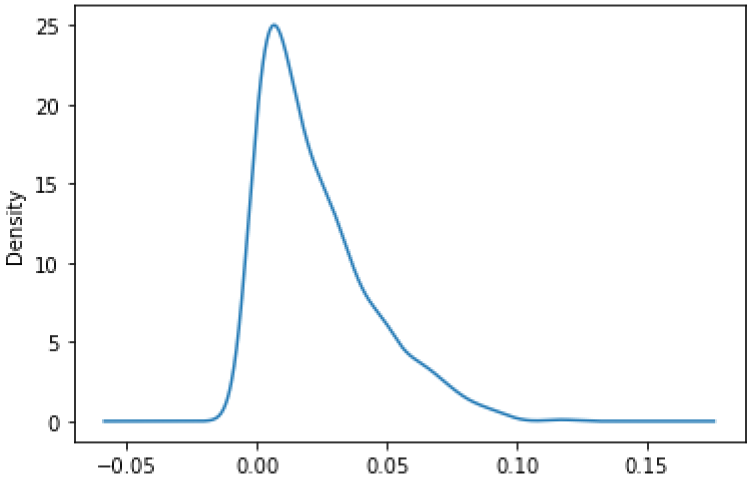
	count	mean	std	min	25%	50%	75%
Pregnancies	768.0	0.022645	0.020956	0.00000	0.006476	0.016716	0.033330
Glucose	768.0	0.682903	0.161166	0.00000	0.587637	0.704501	0.801606
BloodPressure	768.0	0.402801	0.153428	0.00000	0.317522	0.430685	0.511070
SkinThickness	768.0	0.112086	0.092546	0.00000	0.000000	0.114464	0.181524
Insulin	768.0	0.318921	0.338570	0.00000	0.000000	0.249215	0.632833
BMI	768.0	0.186874	0.063402	0.00000	0.146291	0.186167	0.226831
DiabetesPedigreeFunction	768.0	0.002710	0.001902	0.00025	0.001379	0.002159	0.003507
Age	768.0	0.195434	0.080940	0.03246	0.139930	0.181108	0.239313
Outcome	768.0	0.348958	0.476951	0.00000	0.000000	0.000000	1.000000

In [41]:

```
NormalizedX_N.Pregnancies.plot(kind="density")
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1de7739b100>
```

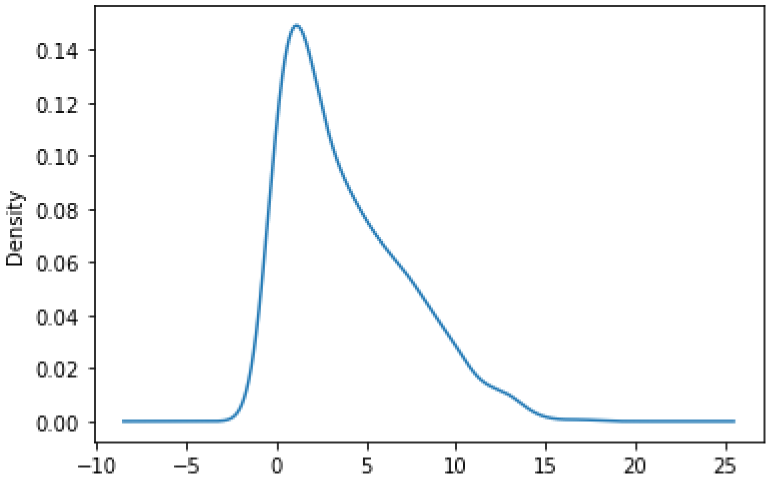


In [42]:

```
df.Pregnancies.plot(kind="density")
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1de7739d640>
```



In [43]:

```
# The normalized data is almost the same of original data in th distribution ,
# so the best use in analysis or model building is the Rescale and stadarndized data
```

In [ ]:

In [ ]:

In [ ]: