LP1Da2

December 10, 2021

[3]:
```
import numpy as np
import pandas as pd
%matplotlib inline

import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```
data=pd.read_csv('Pima.csv')
```

[5]:
```
data.head(5) #########printing data
```

[5]:

	x1	x2	x3	x4	x5	x6	x7	x8	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

[6]:
```
data.shape ##########how many features and rows
```

[6]:
```
(768, 9)
```

[7]:
```
data.info()  ### info about dataset  2.2 Sumarraizing dataset for prediction
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
x1       768 non-null int64
x2       768 non-null int64
x3       768 non-null int64
x4       768 non-null int64
x5       768 non-null int64
x6       768 non-null float64
x7       768 non-null float64
x8       768 non-null int64
class    768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[8]:
```
data['x1'].describe()  ###statistics similary of others
```

[8]:
```
count    768.000000
mean       3.845052
std        3.369578
min        0.000000
25%        1.000000
50%        3.000000
75%        6.000000
max       17.000000
Name: x1, dtype: float64
```

[9]:
```
data.dtypes  ## datatypes
```

[9]:
```
x1         int64
x2         int64
x3         int64
x4         int64
x5         int64
x6       float64
x7       float64
x8         int64
class      int64
dtype: object
```

[8]:
```
train=np.array(data.iloc[0:600])  ##2.1 Loading data into training and testing
test=np.array(data.iloc[600:768])
```

[9]:
```
train.shape  ### traiinging data size
```

[9]:
```
(600, 9)
```

[10]:
```
test.shape ### testing datasize
```

[10]:
```
(168, 9)
```

[11]:
```
from sklearn.naive_bayes import GaussianNB   ####importing guassian model
```

[13]:
```
model = GaussianNB()
```

[15]:
```
model.fit(train[:,0:8], train[:,8])  ##2.3 training the data for prediction
```

[15]:
```
GaussianNB(priors=None)
```

[19]:
```
predicted= model.predict(test[:,0:8])
print(test[:,8])
print(predicted)  ## predicted data
```

```
[ 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.  0.]
[ 0.  0.  0.  1.  1.  0.  1.  0.  1.  0.  0.  1.  1.  0.  1.  0.  0.  0.
  1.  0.  0.  1.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  1.  1.  0.  0.  0.  0.  0.
  0.  1.  0.  1.  1.  0.  1.  1.  1.  1.  0.  0.  0.  0.  0.  1.  1.  0.
  0.  1.  0.  1.  1.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  1.
  0.  1.  0.  1.  0.  1.  0.  0.  0.  0.  0.  0.  1.  1.  0.  0.  0.  0.
  1.  0.  1.  0.  1.  0.  0.  1.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  1.
  1.  1.  1.  0.  1.  0.  0.  0.  0.  1.  1.  1.  1.  0.  0.  1.  0.  1.
  0.  1.  0.  0.  0.  0.]
```

[21]:
```
count=0            ###### calulating acuracy
for l in range(168):
    if(predicted[l]==test[l,8]):
        count=count+1
```

[22]:
```
print(count)  ###### print no of correctly matched samples out of 168
```

```
128
```

[23]:
```
############ Accuracy is


print(count/168)
```

```
0.7619047619047619
```