

LP1  
Assignment 12  
Data Analysis

Title:- Classification

Date of Completion:- 19.11.20

Problem Statement :- Download Pima Indians Diabetes datasets. Use Naive Bayes Algorithm for classification load from the csv file. and split into training & testing set. Summarize the properties in the training dataset so that we can calculate probabilities and make prediction. Classify samples from the test dataset & a summarized training dataset.

Learning Outcome Objectives :- Understand classification & make prediction

Learning Outcome :- Students will be able to understand classification and make predictions.

Software &/Hardware Requirement :- OS (Linux), Python, Pima Indians Diabetes dataset.

Theory:-

Naive Bayes:-

1) Classifier technique

2) It assumes that the presence of a particular feature in a class is related to the presence of any other feature.



- 3) It is easy to build and particularly useful for very large dataset.
4. Along with simplicity it is believed to outperform even sophisticated classification models.

### Bayes Theorem:-

Finds the probability of event occurring given the probability of another event that has already occurred.

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

- 1)  $P(A|B)$  is the posterior probability of class (A, target) given predictor (B, attributes).
- 2)  $P(A)$  is the prior probability of class.
- 3)  $P(B|A)$  is the likelihood which is the probability of predictor given a class.
- 4)  $P(B)$  is the prior probability of predictor.

### Test Cases:-

Input / Case	Actual O/p	Expected O/p	Remark.
Is <del>diabetic</del> diabetic	1 (true)	1 (true)	Passed
is not <del>diabetic</del> diabetic	0	0	Passed

1 → have ~~diabetic~~ diabetic

0 → does not have ~~diabetic~~ diabetic

### Conclusion:-

Thus I classified the given data in 2 parts diabetic and not diabetic using naive bayes model.

## CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
```

```
data=pd.read_csv('Pima.csv')
data.head(5)
data.shape
data.info()
```

```
data['x1'].describe()
```

```
data.dtypes
```

```
train=np.array(data.iloc[0:600])
test=np.array(data.iloc[600:768])
```

```
train.shape
test.shape
```

```
model = GaussianNB()
model.fit(train[:,0:8], train[:,8])
predicted= model.predict(test[:,0:8])
print(test[:,8])
print(predicted)
```

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

```
print("Matched samples:",count)
```

```
print("Accuracy:",(count/168))
```



## OUTPUT

**Matched samples: 128**

**Accuracy: 0.7619047619047619**

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
untitled1.py* x
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.naive_bayes import GaussianNB
6
7 data=pd.read_csv('Pima.csv')
8 data.head(5)
9 data.shape
10 data.info()
11
12 data['x1'].describe()
13
14 data.dtypes
15
16 train=np.array(data.iloc[0:600])
17 test=np.array(data.iloc[600:768])
18
19 train.shape
20 test.shape
21
22 model = GaussianNB()
23 model.fit(train[:,0:8], train[:,8])
24 predicted= model.predict(test[:,0:8])
25 print(test[:,8])
26 print(predicted)
27
28 count=0
29 for l in range(168):
30     if(predicted[l]==test[l,8]):
31         count=count+1
32
33 print("Matched samples:",count)
34
35 print("Accuracy:",(count/168))
36
37
38
```

Python console

```
In [35]: data.head(5)
Out[35]:
```

	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

```
In [36]: data.shape
Out[36]: (768, 9)

In [37]: data.info()
<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

In [38]: data['x1'].describe()
Out[38]:
```

	count	mean	std	min	25%	50%	75%	max
x1	768.000000	3.845052	3.369578	0.000000	1.000000	3.000000	6.000000	17.000000

```
Name: x1, dtype: float64

In [39]: data.dtypes
Out[39]:
```

	x1	x2	x3	x4	x5
	int64	int64	int64	int64	int64

IPython console History log

Permissions: RW

```
IPython console
Console 1/A X
max 17.000000
Name: x1, dtype: float64

In [39]: data.dtypes
Out[39]:
x1      int64
x2      int64
x3      int64
x4      int64
x5      int64
x6      float64
x7      float64
x8      int64
class   int64
dtype: object

In [40]: train=np.array(data.iloc[0:600])
...: test=np.array(data.iloc[600:768])

In [41]: train.shape
Out[41]: (600, 9)

In [42]: test.shape
Out[42]: (168, 9)

In [43]: model = GaussianNB()
...: model.fit(train[:,0:8], train[:,8])
...: predicted= model.predict(test[:,0:8])

In [44]: print(test[:,8])
[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. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 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. 1. 1. 0. 0. 1.
 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 1. 0. 1.]

In [45]: print(predicted)
[0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0.
 1. 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. 0. 0. 0. 1. 0. 1. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0.
 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 1.
 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 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. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0.]
```

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/srushti/BE Sem1/my/LP1/DA/2/untitled1.py
untitled1.py* x
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.naive_bayes import GaussianNB
6
7 data=pd.read_csv('Pima.csv')
8 data.head(5)
9 data.shape
10 data.info()
11
12 data['x1'].describe()
13
14 data.dtypes
15
16 train=np.array(data.iloc[0:600])
17 test=np.array(data.iloc[600:768])
18
19 train.shape
20 test.shape
21
22 model = GaussianNB()
23 model.fit(train[:,0:8], train[:,8])
24 predicted= model.predict(test[:,0:8])
25 print(test[:,8])
26 print(predicted)
27
28 count=0
29 for l in range(168):
30     if(predicted[l]==test[l,8]):
31         count=count+1
32
33 print("Matched samples:",count)
34
35 print("Accuracy:",(count/168))
36
37
38
IPython console
In [44]: print(test[:,8])
[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. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0.
 0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 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. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1.]

In [45]: print(predicted)
[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. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1.
 1. 0. 0. 0. 0. 0. 0. 1. 0. 1. 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. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 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. 1. 0. 0. 1.
 1. 1. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0.]

In [46]: count=0
...: for l in range(168):
...:     if(predicted[l]==test[l,8]):
...:         count=count+1

In [47]: print("Matched samples:",count)
...:
...: print("Accuracy:",(count/168))
Matched samples: 128
Accuracy: 0.7619047619047619

In [48]:
Permissions: RW End-of-lines: LF Encod
```