LP1

Assignment C2

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~~ a csv file. and split into training & testing set. Summerize the properties in the training dataset so that we can calculate probabilities and make prediction Classify Samples from a test dataset & a summerized 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 out perform even sofisticated classification models.

Bayes Theorem:-
Finds the probability of event occuring given the probability of another event that has already occured.

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

1) P(A/B) is the proterior 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 diabitic | 1 (true) | 1 (true) | Passed |
| is not diabitic | 0 | 0 | Passed |

1 → have diabitic
0 → does not have diabitic

Page :
Date : / /

Conclusion:-

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

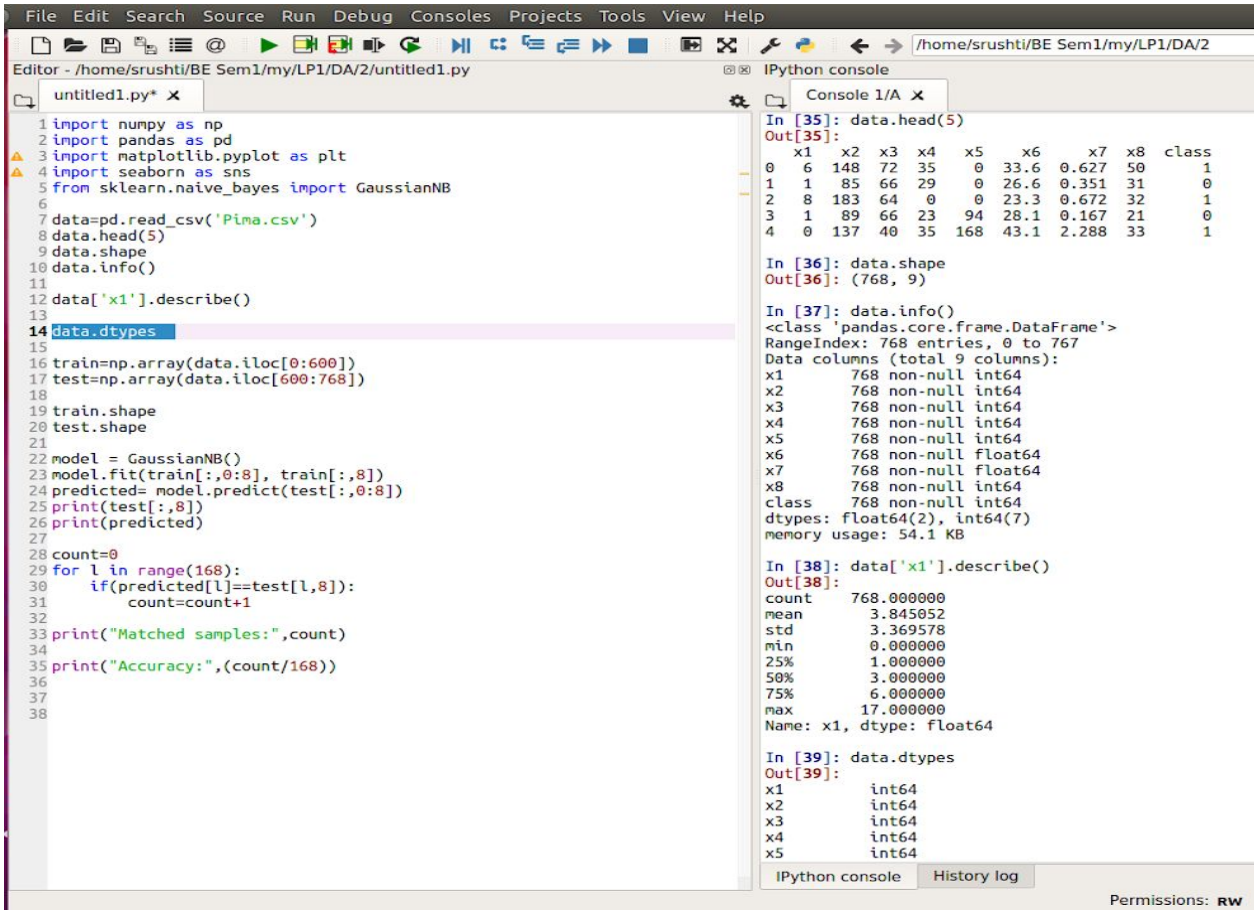
# CODE

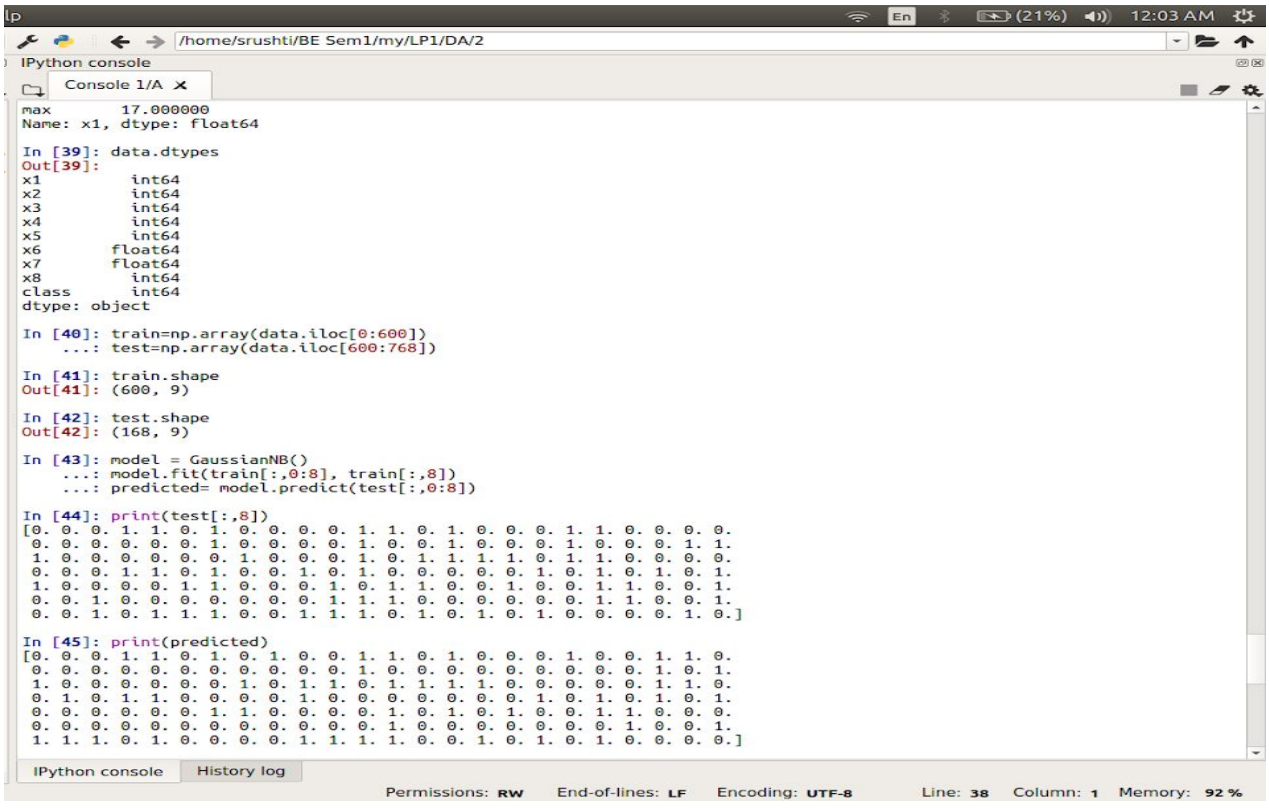
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB

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))
```

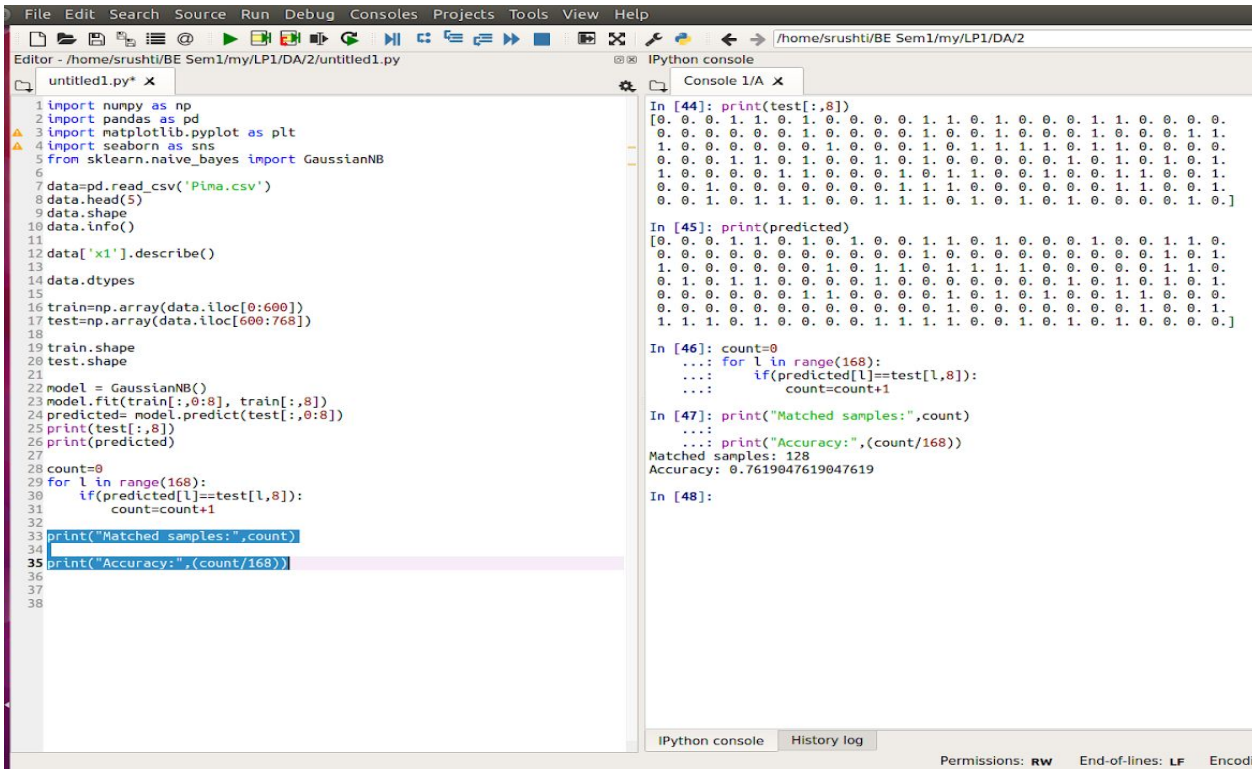
# OUTPUT

## Matched samples: 128

## Accuracy: 0.7619047619047619




File Edit Search Source Run Debug Consoles Projects Tools View Help
/home/srushti/BE Sem1/my/LP1/DA/2
Editor - /home/srushti/BE Sem1/my/LP1/DA/2/untitled1.py
untitled1.py*
IPython console
Console 1/A
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))
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. 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. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 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.]
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. 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. 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. 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.]
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]:
IPython console
History log
Permissions: RW
End-of-lines: LF
Encodi