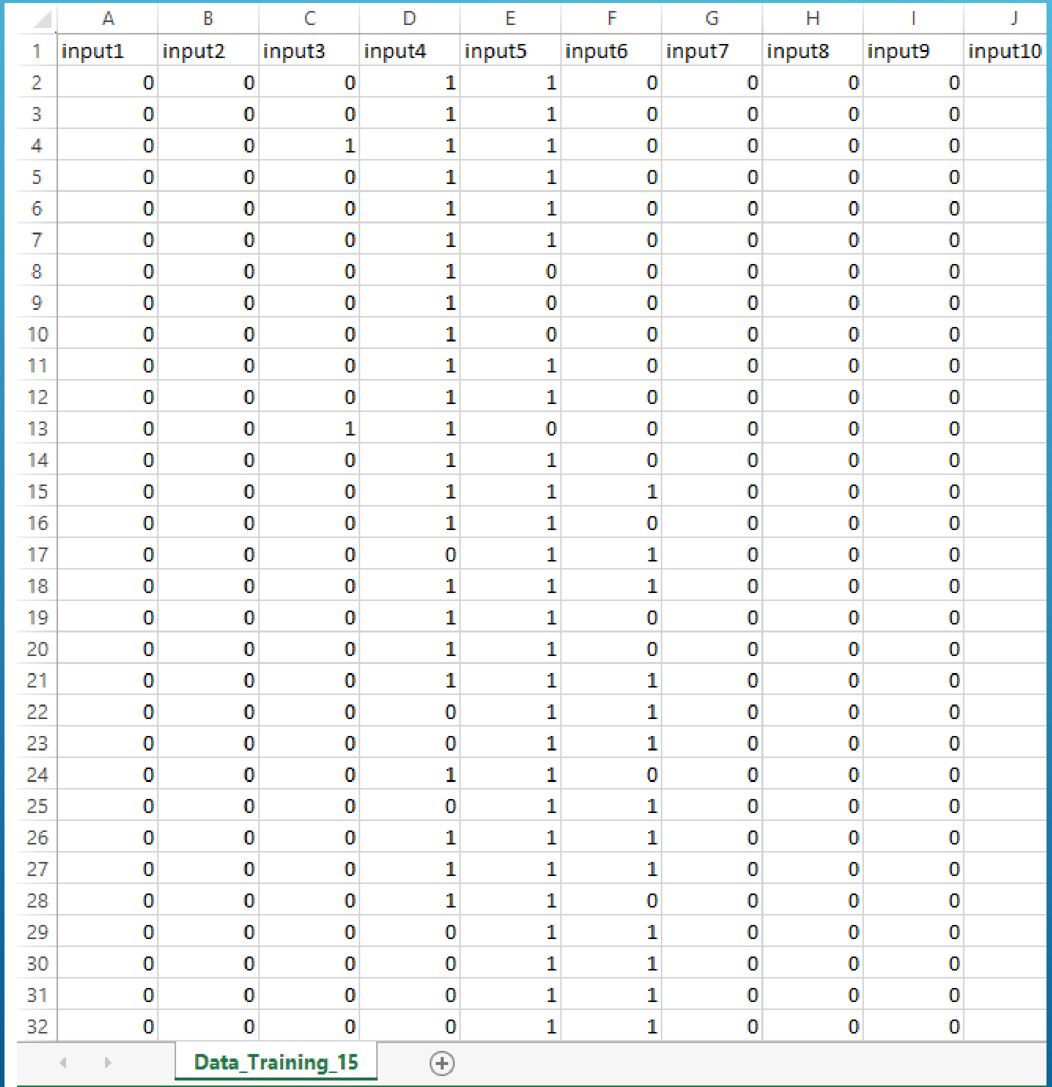
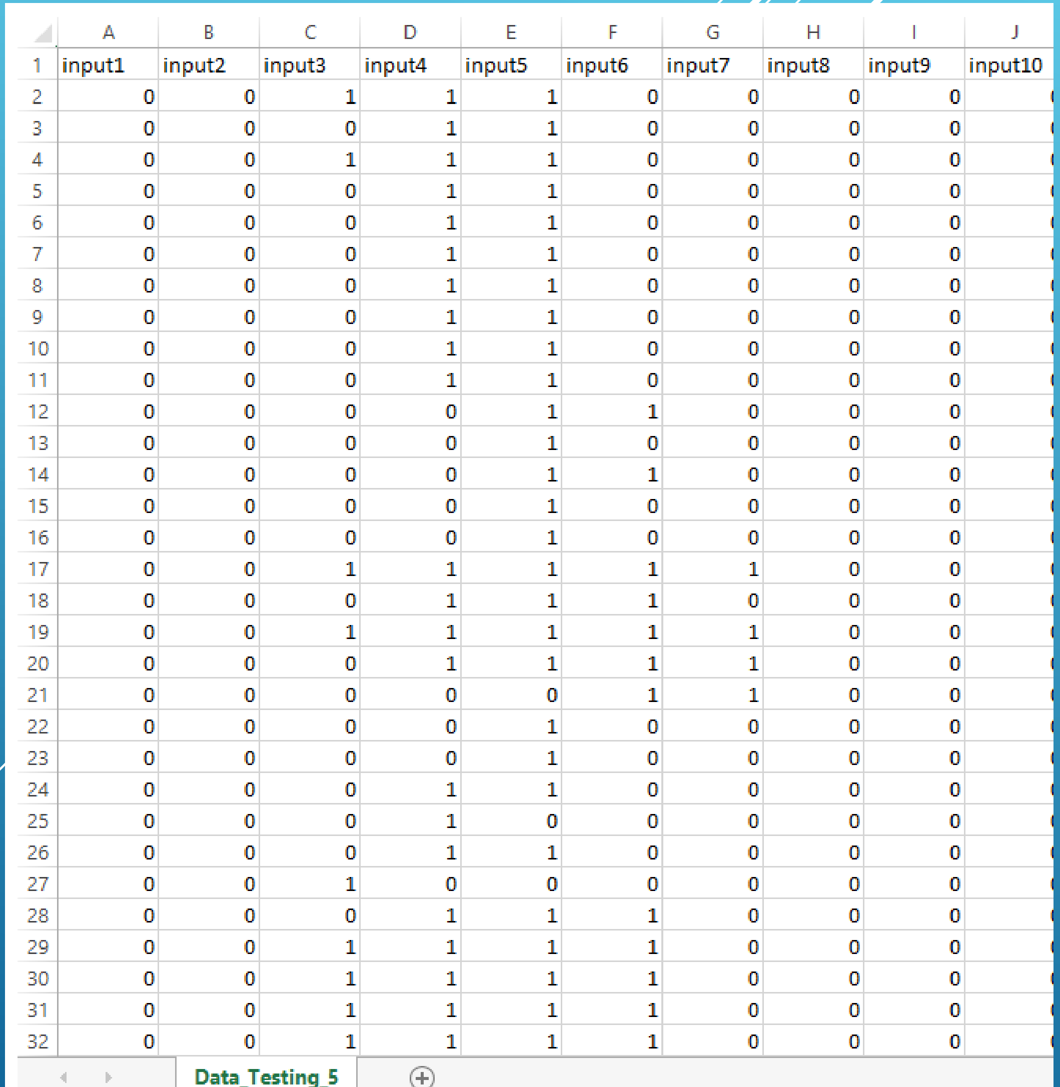
TASK 5 NAIVE BAYES

1. Get The Dataset Number 8x8



	A	B	C	D	E	F	G	H	I	J
1	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10
2	0	0	0	1	1	0	0	0	0	
3	0	0	0	1	1	0	0	0	0	
4	0	0	1	1	1	0	0	0	0	
5	0	0	0	1	1	0	0	0	0	
6	0	0	0	1	1	0	0	0	0	
7	0	0	0	1	1	0	0	0	0	
8	0	0	0	1	0	0	0	0	0	
9	0	0	0	1	0	0	0	0	0	
10	0	0	0	1	0	0	0	0	0	
11	0	0	0	1	1	0	0	0	0	
12	0	0	0	1	1	0	0	0	0	
13	0	0	1	1	0	0	0	0	0	
14	0	0	0	1	1	0	0	0	0	
15	0	0	0	1	1	1	0	0	0	
16	0	0	0	1	1	0	0	0	0	
17	0	0	0	0	1	1	0	0	0	
18	0	0	0	1	1	1	0	0	0	
19	0	0	0	1	1	0	0	0	0	
20	0	0	0	1	1	0	0	0	0	
21	0	0	0	1	1	1	0	0	0	
22	0	0	0	0	1	1	0	0	0	
23	0	0	0	0	1	1	0	0	0	
24	0	0	0	1	1	0	0	0	0	
25	0	0	0	0	1	1	0	0	0	
26	0	0	0	1	1	1	0	0	0	
27	0	0	0	1	1	1	0	0	0	
28	0	0	0	1	1	0	0	0	0	
29	0	0	0	0	1	1	0	0	0	
30	0	0	0	0	1	1	0	0	0	
31	0	0	0	0	1	1	0	0	0	
32	0	0	0	0	1	1	0	0	0	

Data_Training_15

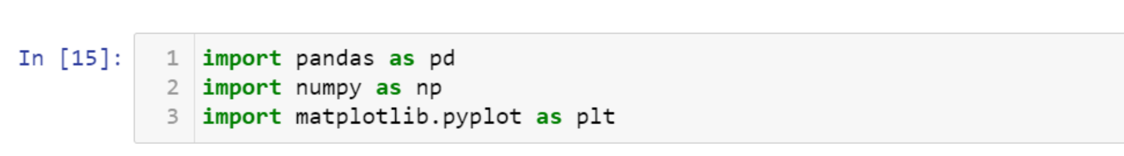


	A	B	C	D	E	F	G	H	I	J
1	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10
2	0	0	1	1	1	0	0	0	0	
3	0	0	0	1	1	0	0	0	0	
4	0	0	1	1	1	0	0	0	0	
5	0	0	0	1	1	0	0	0	0	
6	0	0	0	1	1	0	0	0	0	
7	0	0	0	1	1	0	0	0	0	
8	0	0	0	1	1	0	0	0	0	
9	0	0	0	1	1	0	0	0	0	
10	0	0	0	1	1	0	0	0	0	
11	0	0	0	1	1	0	0	0	0	
12	0	0	0	0	1	1	0	0	0	
13	0	0	0	0	1	0	0	0	0	
14	0	0	0	0	1	1	0	0	0	
15	0	0	0	0	1	0	0	0	0	
16	0	0	0	0	1	0	0	0	0	
17	0	0	1	1	1	1	1	0	0	
18	0	0	0	1	1	1	0	0	0	
19	0	0	1	1	1	1	1	0	0	
20	0	0	0	1	1	1	1	0	0	
21	0	0	0	0	0	1	1	0	0	
22	0	0	0	0	1	0	0	0	0	
23	0	0	0	0	1	0	0	0	0	
24	0	0	0	1	1	0	0	0	0	
25	0	0	0	1	0	0	0	0	0	
26	0	0	0	1	1	0	0	0	0	
27	0	0	1	0	0	0	0	0	0	
28	0	0	0	1	1	1	0	0	0	
29	0	0	1	1	1	1	0	0	0	
30	0	0	1	1	1	1	0	0	0	
31	0	0	1	1	1	1	0	0	0	
32	0	0	1	1	1	1	0	0	0	

Data_Testing_5

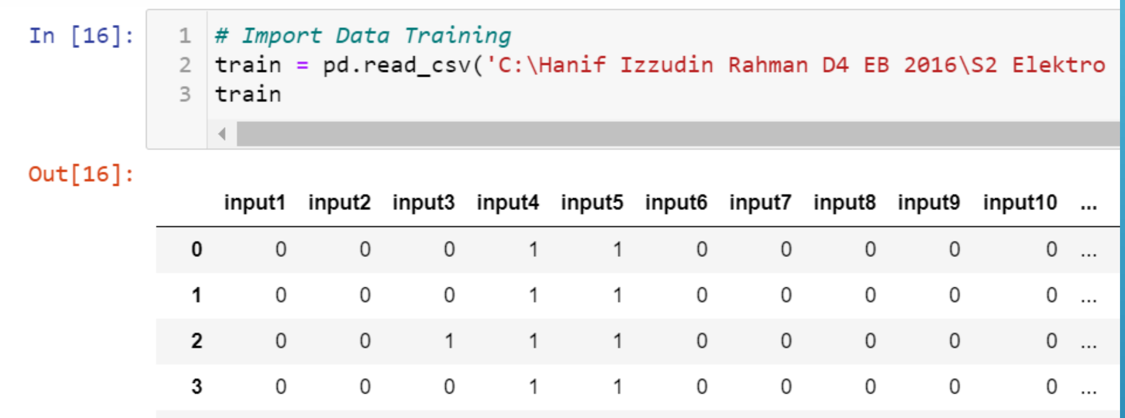
CODING WITH PYTHON

3. Import Libraries Needed



```
In [15]:  import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
```

4. Import Dataset for Training (150 Data)

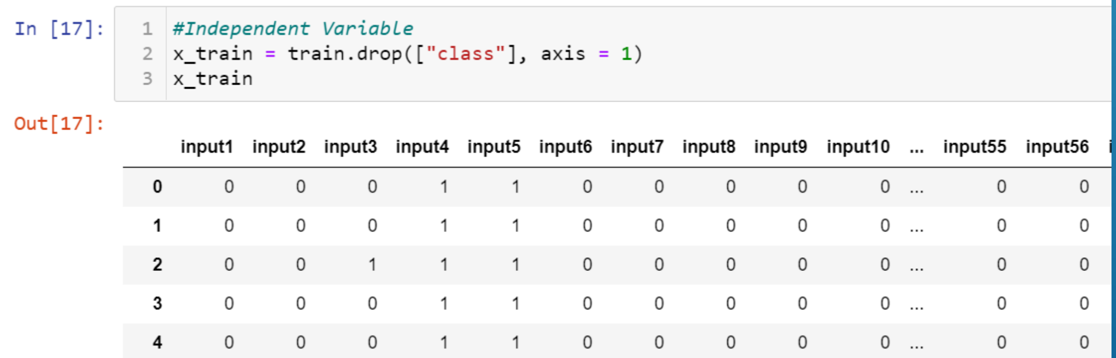


```
In [16]:  # Import Data Training
          train = pd.read_csv('C:\Hanif Izzudin Rahman D4 EB 2016\S2 Elektro
          train
```

Out[16]:

	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	...
0	0	0	0	1	1	0	0	0	0	0	...
1	0	0	0	1	1	0	0	0	0	0	...
2	0	0	1	1	1	0	0	0	0	0	...
3	0	0	0	1	1	0	0	0	0	0	...

5. Get Input for Dataset Training

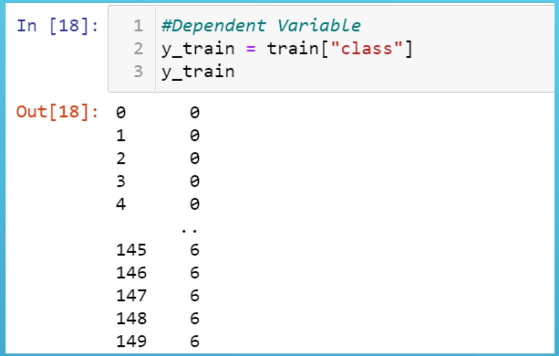


```
In [17]:  #Independent Variable
          x_train = train.drop(["class"], axis = 1)
          x_train
```

Out[17]:

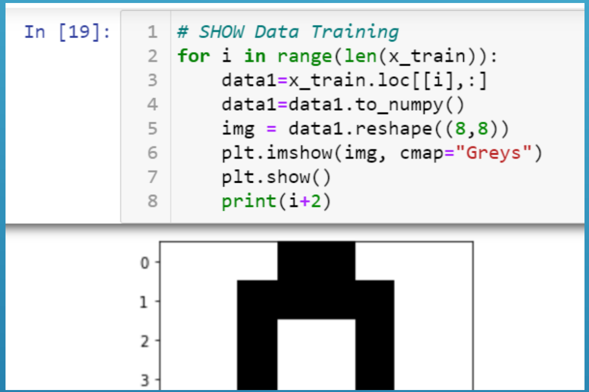
	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	...	input55	input56
0	0	0	0	1	1	0	0	0	0	0	...	0	0
1	0	0	0	1	1	0	0	0	0	0	...	0	0
2	0	0	1	1	1	0	0	0	0	0	...	0	0
3	0	0	0	1	1	0	0	0	0	0	...	0	0
4	0	0	0	1	1	0	0	0	0	0	...	0	0

6. Get Output for Dataset Training



```
In [18]:  #Dependent Variable
          y_train = train["class"]
          y_train

Out[18]:  0      0
          1      0
          2      0
          3      0
          4      0
                ..
          145    6
          146    6
          147    6
          148    6
          149    6
```

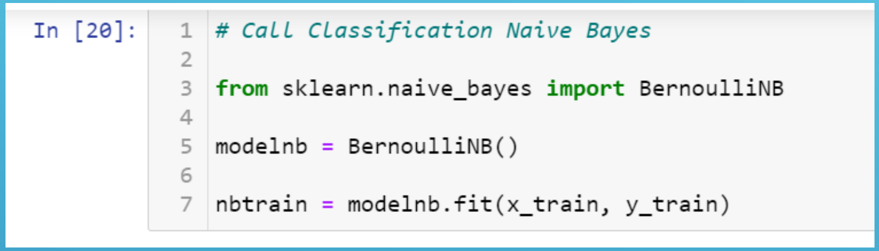
7. Show Dataset Training (Optional)



```
In [19]:  # SHOW Data Training
          for i in range(len(x_train)):
              data1=x_train.loc[[i],:]
              data1=data1.to_numpy()
              img = data1.reshape((8,8))
              plt.imshow(img, cmap="Greys")
              plt.show()
              print(i+2)
```

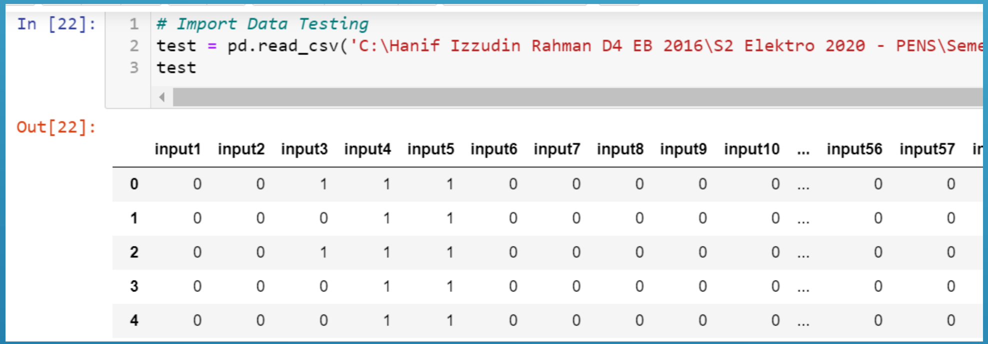
CODING WITH PYTHON

8. Call Classification Bernoulli Naïve Bayes



```
In [20]:  # Call Classification Naive Bayes

          from sklearn.naive_bayes import BernoulliNB

          modelnb = BernoulliNB()

          nbtrain = modelnb.fit(x_train, y_train)
```

9. Import Dataset for Testing (50 Data)

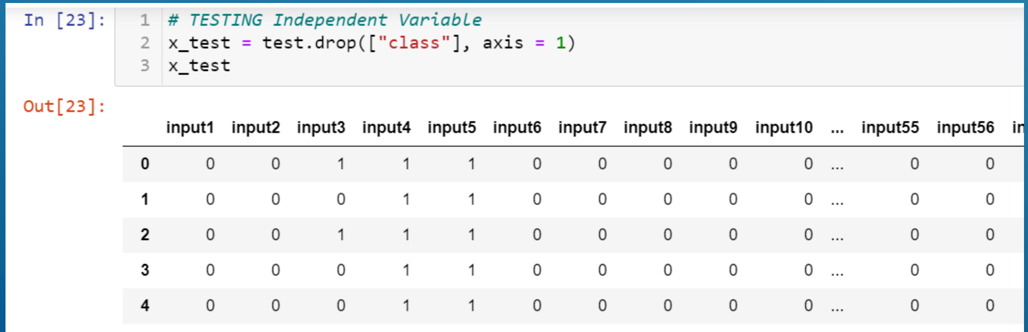


```
In [22]:  # Import Data Testing
          test = pd.read_csv('C:\Hanif Izzudin Rahman D4 EB 2016\S2 Elektro 2020 - PENS\Seme
          test
```

Out[22]:

	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	...	input56	input57	ir
0	0	0	1	1	1	0	0	0	0	0	...	0	0	
1	0	0	0	1	1	0	0	0	0	0	...	0	0	
2	0	0	1	1	1	0	0	0	0	0	...	0	0	
3	0	0	0	1	1	0	0	0	0	0	...	0	0	
4	0	0	0	1	1	0	0	0	0	0	...	0	0	

10. Get Input for Dataset Testing

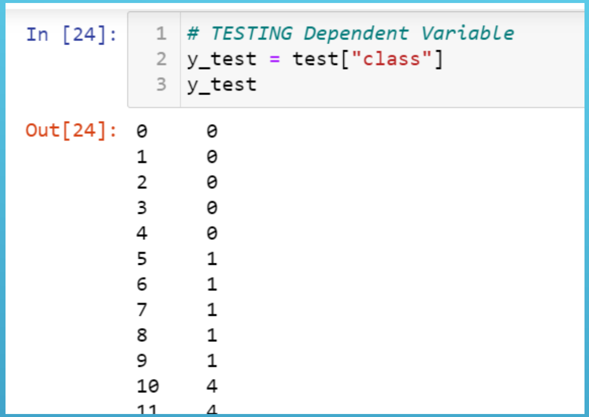


```
In [23]:  # TESTING Independent Variable
          x_test = test.drop(["class"], axis = 1)
          x_test
```

Out[23]:

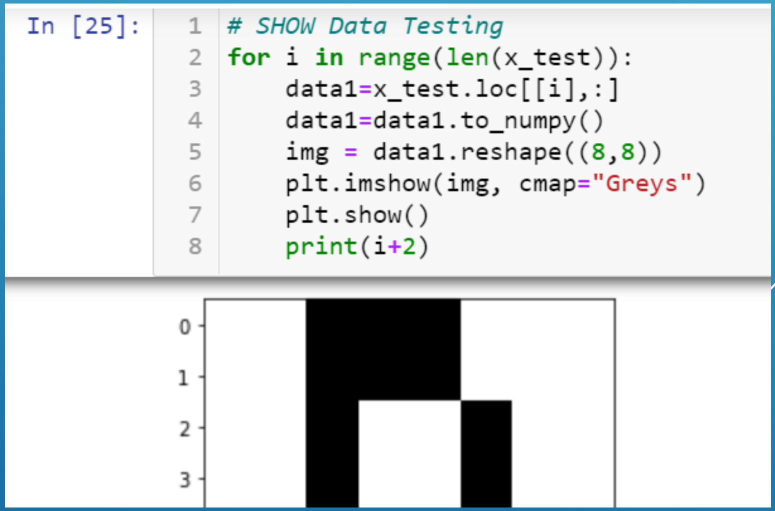
	input1	input2	input3	input4	input5	input6	input7	input8	input9	input10	...	input55	input56	in
0	0	0	1	1	1	0	0	0	0	0	...	0	0	
1	0	0	0	1	1	0	0	0	0	0	...	0	0	
2	0	0	1	1	1	0	0	0	0	0	...	0	0	
3	0	0	0	1	1	0	0	0	0	0	...	0	0	
4	0	0	0	1	1	0	0	0	0	0	...	0	0	

11. Get Output for Dataset Testing



```
In [24]:  # TESTING Dependent Variable
          y_test = test["class"]
          y_test

Out[24]:  0     0
          1     0
          2     0
          3     0
          4     0
          5     1
          6     1
          7     1
          8     1
          9     1
          10    4
          11    4
```

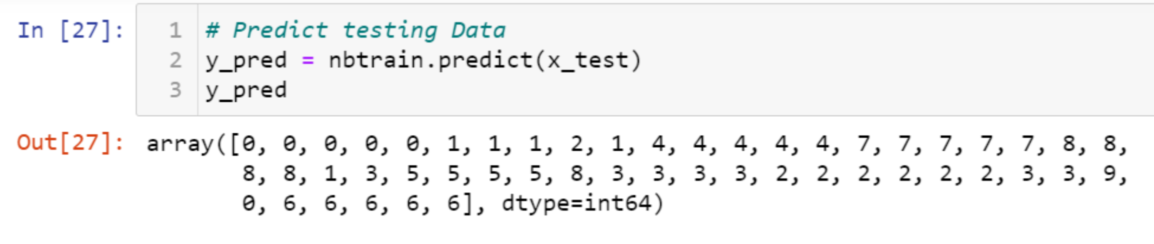
12. Show Dataset Training (Optional)



```
In [25]:  # SHOW Data Testing
          for i in range(len(x_test)):
              data1=x_test.loc[[i],:]
              data1=data1.to_numpy()
              img = data1.reshape((8,8))
              plt.imshow(img, cmap="Greys")
              plt.show()
              print(i+2)
```

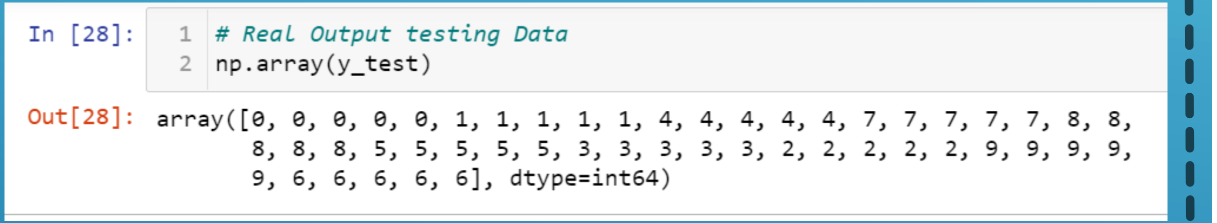
CODING WITH PYTHON

13. Predict Testing Data



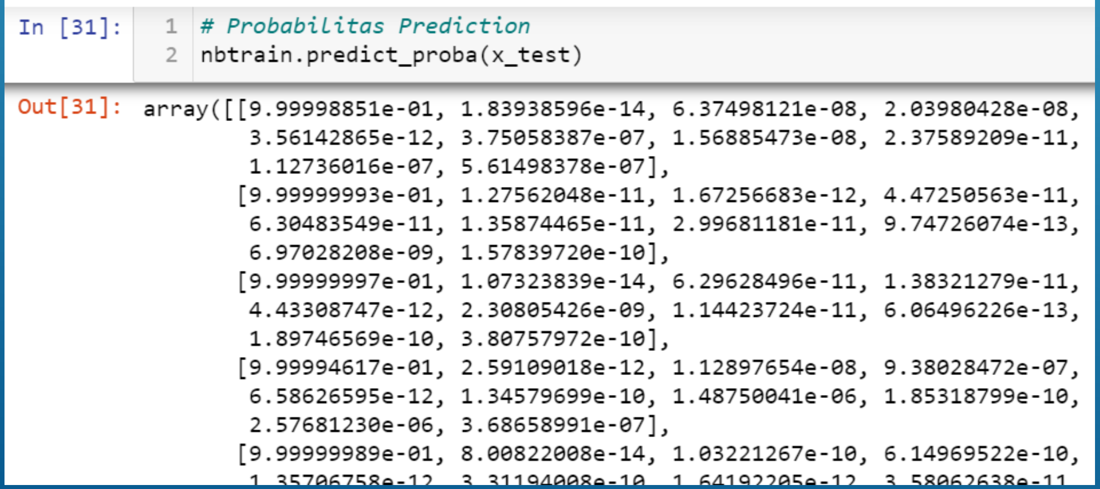
```
In [27]:  # Predict testing Data
          y_pred = nbtrain.predict(x_test)
          y_pred

Out[27]: array([0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 4, 4, 4, 4, 4, 7, 7, 7, 7, 7, 8, 8,
                8, 8, 1, 3, 5, 5, 5, 5, 8, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 3, 3, 9,
                0, 6, 6, 6, 6, 6], dtype=int64)
```

14. Real Output Testing Data



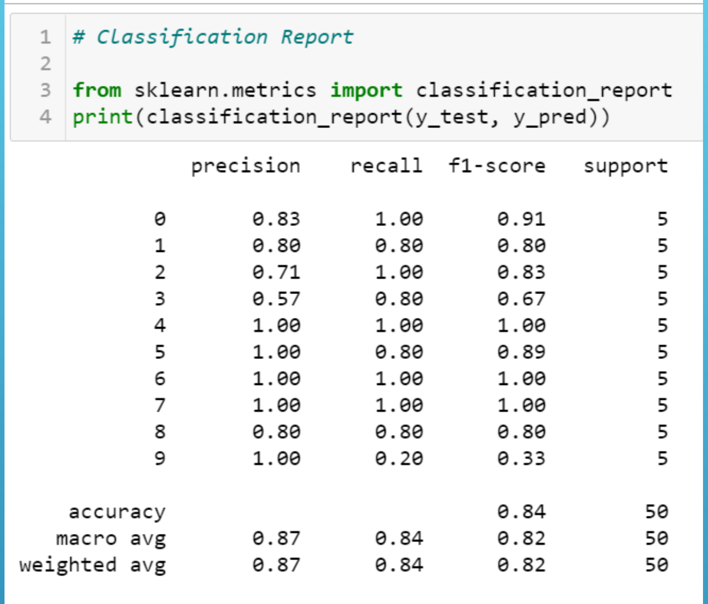
```
In [28]:  # Real Output testing Data
          np.array(y_test)

Out[28]: array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 4, 4, 4, 4, 4, 7, 7, 7, 7, 7, 8, 8,
                8, 8, 8, 5, 5, 5, 5, 5, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 9, 9, 9, 9,
                9, 6, 6, 6, 6, 6], dtype=int64)
```

15. Show Probability of Prediction



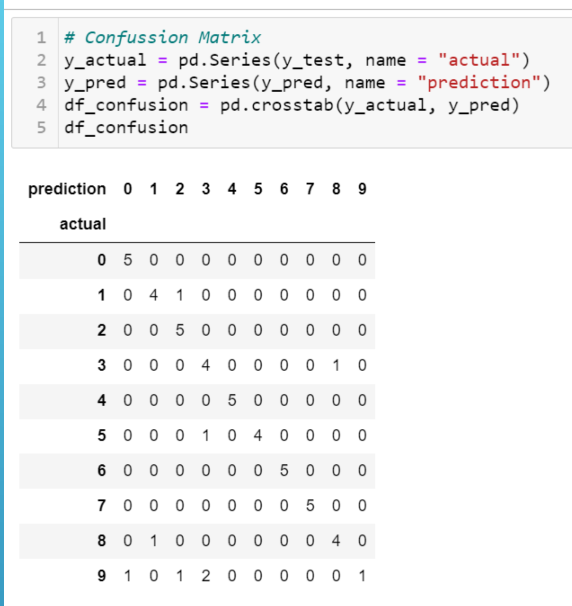
```
In [31]:  # Probabilitas Prediction
          nbtrain.predict_proba(x_test)

Out[31]: array([[9.99998851e-01, 1.83938596e-14, 6.37498121e-08, 2.03980428e-08,
                 3.56142865e-12, 3.75058387e-07, 1.56885473e-08, 2.37589209e-11,
                 1.12736016e-07, 5.61498378e-07],
                [9.99999993e-01, 1.27562048e-11, 1.67256683e-12, 4.47250563e-11,
                 6.30483549e-11, 1.35874465e-11, 2.99681181e-11, 9.74726074e-13,
                 6.97028208e-09, 1.57839720e-10],
                [9.99999997e-01, 1.07323839e-14, 6.29628496e-11, 1.38321279e-11,
                 4.43308747e-12, 2.30805426e-09, 1.14423724e-11, 6.06496226e-13,
                 1.89746569e-10, 3.80757972e-10],
                [9.99994617e-01, 2.59109018e-12, 1.12897654e-08, 9.38028472e-07,
                 6.58626595e-12, 1.34579699e-10, 1.48750041e-06, 1.85318799e-10,
                 2.57681230e-06, 3.68658991e-07],
                [9.99999989e-01, 8.00822008e-14, 1.03221267e-10, 6.14969522e-10,
                 1.35706758e-12, 3.31194008e-10, 1.64192205e-12, 3.58062638e-11,
```

16. Classification Report & Confusion Matrix



```python
# Classification Report

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       0.80      0.80      0.80         5
           2       0.71      1.00      0.83         5
           3       0.57      0.80      0.67         5
           4       1.00      1.00      1.00         5
           5       1.00      0.80      0.89         5
           6       1.00      1.00      1.00         5
           7       1.00      1.00      1.00         5
           8       0.80      0.80      0.80         5
           9       1.00      0.20      0.33         5

    accuracy                           0.84        50
   macro avg       0.87      0.84      0.82        50
weighted avg       0.87      0.84      0.82        50
```



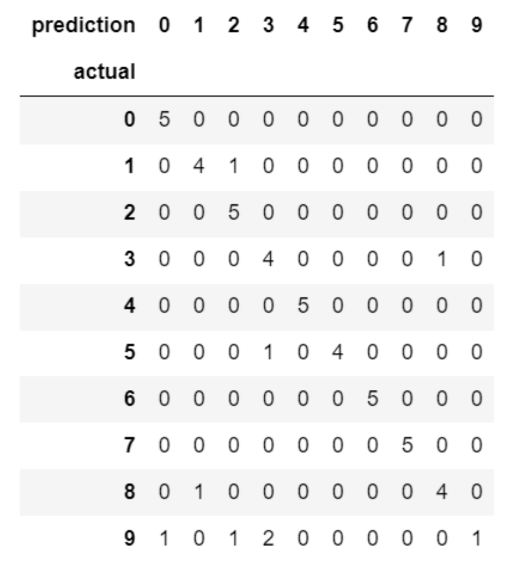
```python
# Confussion Matrix
y_actual = pd.Series(y_test, name = "actual")
y_pred = pd.Series(y_pred, name = "prediction")
df_confusion = pd.crosstab(y_actual, y_pred)
df_confusion
```

prediction	0	1	2	3	4	5	6	7	8	9
actual										
0	5	0	0	0	0	0	0	0	0	0
1	0	4	1	0	0	0	0	0	0	0
2	0	0	5	0	0	0	0	0	0	0
3	0	0	0	4	0	0	0	0	1	0
4	0	0	0	0	5	0	0	0	0	0
5	0	0	0	1	0	4	0	0	0	0
6	0	0	0	0	0	0	5	0	0	0
7	0	0	0	0	0	0	0	5	0	0
8	0	1	0	0	0	0	0	0	4	0
9	1	0	1	2	0	0	0	0	0	1

DATA REPORT
(DATA TRAINING – DATA TESTING)

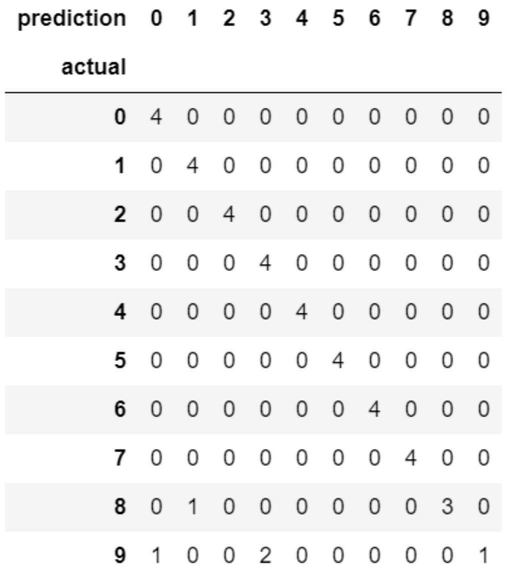
Confusion Matrix

(15 - 5)



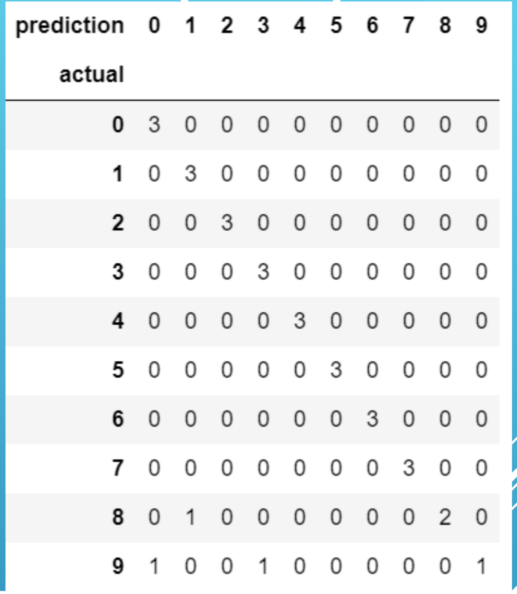
prediction actual	0	1	2	3	4	5	6	7	8	9
0	5	0	0	0	0	0	0	0	0	0
1	0	4	1	0	0	0	0	0	0	0
2	0	0	5	0	0	0	0	0	0	0
3	0	0	0	4	0	0	0	0	1	0
4	0	0	0	0	5	0	0	0	0	0
5	0	0	0	1	0	4	0	0	0	0
6	0	0	0	0	0	0	5	0	0	0
7	0	0	0	0	0	0	0	5	0	0
8	0	1	0	0	0	0	0	0	4	0
9	1	0	1	2	0	0	0	0	0	1

(16 – 4)



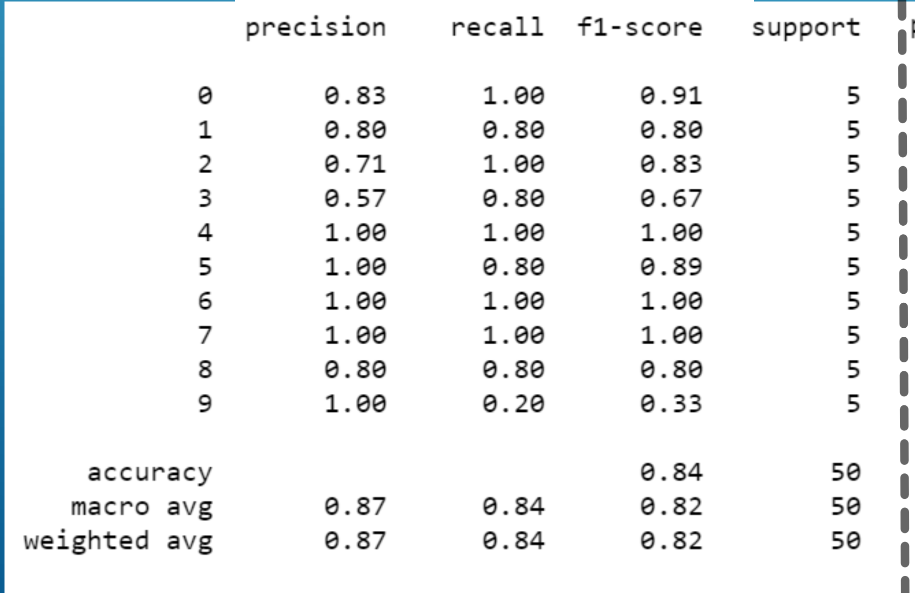
prediction actual	0	1	2	3	4	5	6	7	8	9
0	4	0	0	0	0	0	0	0	0	0
1	0	4	0	0	0	0	0	0	0	0
2	0	0	4	0	0	0	0	0	0	0
3	0	0	0	4	0	0	0	0	0	0
4	0	0	0	0	4	0	0	0	0	0
5	0	0	0	0	0	4	0	0	0	0
6	0	0	0	0	0	0	4	0	0	0
7	0	0	0	0	0	0	0	4	0	0
8	0	1	0	0	0	0	0	0	3	0
9	1	0	0	2	0	0	0	0	0	1

(17 – 3)

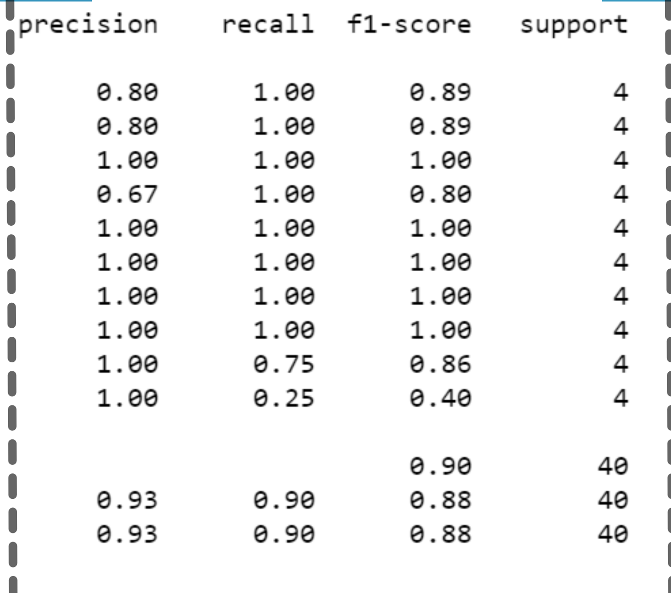


prediction actual	0	1	2	3	4	5	6	7	8	9
0	3	0	0	0	0	0	0	0	0	0
1	0	3	0	0	0	0	0	0	0	0
2	0	0	3	0	0	0	0	0	0	0
3	0	0	0	3	0	0	0	0	0	0
4	0	0	0	0	3	0	0	0	0	0
5	0	0	0	0	0	3	0	0	0	0
6	0	0	0	0	0	0	3	0	0	0
7	0	0	0	0	0	0	0	3	0	0
8	0	1	0	0	0	0	0	0	2	0
9	1	0	0	1	0	0	0	0	0	1

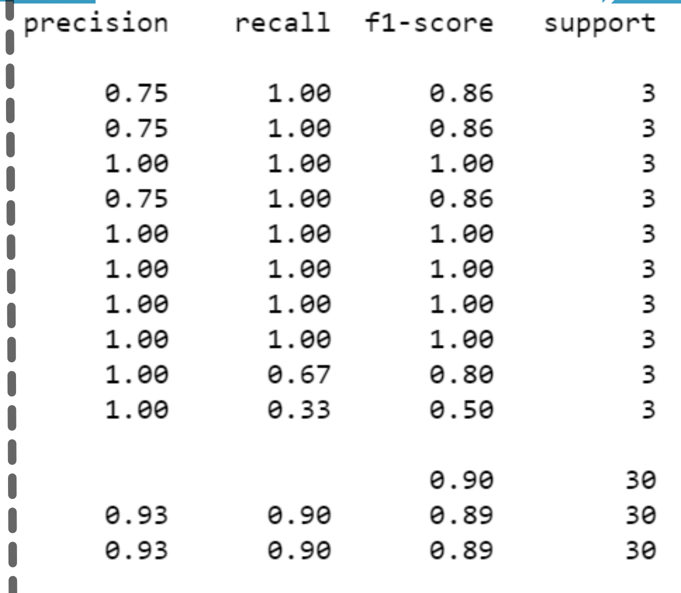
Classification Report



```
              precision    recall  f1-score   support

           0       0.83      1.00      0.91         5
           1       0.80      0.80      0.80         5
           2       0.71      1.00      0.83         5
           3       0.57      0.80      0.67         5
           4       1.00      1.00      1.00         5
           5       1.00      0.80      0.89         5
           6       1.00      1.00      1.00         5
           7       1.00      1.00      1.00         5
           8       0.80      0.80      0.80         5
           9       1.00      0.20      0.33         5

    accuracy                           0.84        50
   macro avg       0.87      0.84      0.82        50
weighted avg       0.87      0.84      0.82        50
```



```
precision    recall  f1-score   support

     0.80      1.00      0.89         4
     0.80      1.00      0.89         4
     1.00      1.00      1.00         4
     0.67      1.00      0.80         4
     1.00      1.00      1.00         4
     1.00      1.00      1.00         4
     1.00      1.00      1.00         4
     1.00      1.00      1.00         4
     1.00      0.75      0.86         4
     1.00      0.25      0.40         4

                         0.90        40
     0.93      0.90      0.88        40
     0.93      0.90      0.88        40
```



```
precision    recall  f1-score   support

     0.75      1.00      0.86         3
     0.75      1.00      0.86         3
     1.00      1.00      1.00         3
     0.75      1.00      0.86         3
     1.00      1.00      1.00         3
     1.00      1.00      1.00         3
     1.00      1.00      1.00         3
     1.00      1.00      1.00         3
     1.00      0.67      0.80         3
     1.00      0.33      0.50         3

                         0.90        30
     0.93      0.90      0.89        30
     0.93      0.90      0.89        30
```

ANALYSIS

- The library naive bayes that we are using here is scikit-learn libraries. There are 5 kind of naive bayes, and we are using Bernoulli Naive Bayes, because the input is a binary number (“0” or “1”)
- We are using 3 experiments, 15 Data Training – 5 Data Testing, 16 Data Training – 4 Data Testing and 17 Data Training – 3 Data Testing. The error of each experiments are different. When the amount of data training is added and the data testing is subtracted, the model is become better, so the error is less.

Number	Error (Training – Testing)					
	15 - 5		16 - 4		17 - 3	
	True	False	True	False	True	False
Number “0”	5	-	4	-	3	-
Number “1”	4	1	4	-	3	-
Number “2”	5	-	4	-	3	-
Number “3”	4	1	4	-	3	-
Number “4”	5	-	4	-	3	-
Number “5”	4	1	4	-	3	-
Number “6”	5	-	4	-	3	-
Number “7”	5	-	4	-	3	-
Number “8”	4	1	3	1	2	1
Number “9”	1	4	1	3	1	2
Total Error	16%		10%		10%	