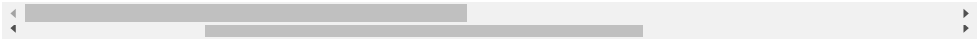


df

alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color
15.6	127.0	2.80	3.06	0.28	2.29	
11.2	100.0	2.65	2.76	0.26	1.28	
18.6	101.0	2.80	3.24	0.30	2.81	
16.8	113.0	3.85	3.49	0.24	2.18	

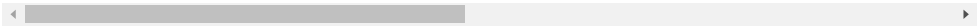
df[df['target']==0].head()

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.28



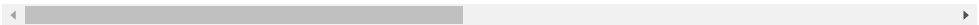
df[df['target']==1].head()

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols
59	12.37	0.94	1.36	10.6	88.0	1.98	0.57	0.26
60	12.33	1.10	2.28	16.0	101.0	2.05	1.09	0.26
61	12.64	1.36	2.02	16.8	100.0	2.02	1.41	0.26
62	13.67	1.25	1.92	18.0	94.0	2.10	1.79	0.26
63	12.37	1.13	2.16	19.0	87.0	3.50	3.10	0.26



df[df['target']==2].head()

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols
130	12.86	1.35	2.32	18.0	122.0	1.51	1.25	0.26
131	12.88	2.99	2.40	20.0	104.0	1.30	1.22	0.26
132	12.81	2.31	2.40	24.0	98.0	1.15	1.09	0.26
133	12.70	3.55	2.36	21.5	106.0	1.70	1.20	0.26
134	12.51	1.24	2.25	17.5	85.0	2.00	0.58	0.26



```
x = df.drop(['target'], axis='columns')
y = df.target
```

x

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	non-hydroxyphenols
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.1
1	16.64	1.76	2.71	11.1	206.0	3.66	3.17	0.26
2	10.34	1.81	2.81	17.0	167.0	3.76	3.44	0.1
3	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
4	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
5	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
6	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
7	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
8	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
9	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
10	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
11	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
12	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
13	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
14	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
15	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
16	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
17	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
18	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
19	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
20	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
21	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
22	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
23	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
24	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
25	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
26	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
27	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
28	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
29	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
30	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
31	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
32	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
33	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
34	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
35	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
36	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
37	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
38	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
39	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
40	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
41	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
42	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
43	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
44	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
45	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
46	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
47	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
48	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
49	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
50	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
51	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
52	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
53	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
54	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
55	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
56	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
57	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
58	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
59	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
60	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
61	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
62	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
63	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
64	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
65	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
66	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
67	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
68	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
69	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
70	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
71	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
72	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
73	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
74	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
75	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
76	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
77	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
78	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
79	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
80	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
81	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
82	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
83	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
84	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
85	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
86	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
87	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
88	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
89	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
90	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
91	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
92	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
93	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
94	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
95	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
96	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
97	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
98	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1
99	16.64	1.81	2.81	17.0	167.0	3.76	3.44	0.1

train test split

```

from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.3)

from sklearn.naive_bayes import GaussianNB, MultinomialNB
model1 = GaussianNB()
model2 = MultinomialNB()
model1.fit(xtrain,ytrain)
model2.fit(xtrain,ytrain)

MultinomialNB()

#GaussianNB
#Accuracy
model1.score(xtest,ytest)

0.9629629629629629

#MultinomialNB
#Accuracy
model2.score(xtest,ytest)

0.8148148148148148

#cross validation
from sklearn.model_selection import cross_val_score
s1 = cross_val_score(GaussianNB(),xtrain,ytrain)

s1

array([1.         , 1.         , 1.         , 0.88        , 0.95833333])

#Average
np.average(s1)

0.9676666666666666

s2 = cross_val_score(MultinomialNB(),xtrain,ytrain)
s2

array([0.8 , 0.96, 0.92, 0.88, 0.75])

#Average
np.average(s2)

0.8620000000000001

#GaussianNB is the best working

```

✓ 0s completed at 11:01 AM

