In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree  import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
mushroom=pd.read_csv('mushrooms.csv')
```

In [3]:

```
mushroom
#observed that class is the output
```

Out[3]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring numbe
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
8119	e	k	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o
8120	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	n	o
8121	e	f	s	n	f	n	a	c	b	n	...	s	o	o	p	o	o
8122	p	k	y	n	f	y	f	c	n	b	...	k	w	w	p	w	o
8123	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o

8124 rows × 23 columns

In [4]:

```
mushroom.describe()
```

Out[4]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	r num
count	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	...	8124	8124	8124	8124	8124	8
unique	2	6	4	10	2	9	2	2	2	12	...	4	9	9	1	4	
top	e	x	y	n	f	n	f	c	b	b	...	s	w	w	p	w	
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	...	4936	4464	4384	8124	7924	7

4 rows × 23 columns

In [5]:

```
mushroom.dtypes
```

Out[5]:

```
class                       object
cap-shape                   object
cap-surface                 object
cap-color                   object
bruises                     object
odor                        object
gill-attachment             object
gill-spacing                object
gill-size                   object
gill-color                  object
stalk-shape                 object
stalk-root                  object
stalk-surface-above-ring    object
stalk-surface-below-ring    object
stalk-color-above-ring      object
stalk-color-below-ring      object
veil-type                   object
veil-color                  object
ring-number                 object
ring-type                   object
spore-print-color           object
population                  object
habitat                     object
dtype: object
```

In [6]:

```
#converting categorical data in numerical data
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
list1=['class','cap-shape','cap-surface','cap-color','bruises','odor','gill-attachment','gill-spaci
ng','gill-size','gill-color','stalk-shape','stalk-root','stalk-surface-above-ring','stalk-surface-
below-ring','stalk-color-above-ring','stalk-color-below-ring','veil-type','veil-color','ring-number
','ring-type','spore-print-color','population','habitat']
for val in list1:
    mushroom[val]=le.fit_transform(mushroom[val].astype(str))
```

In [7]:

```
mushroom
```

Out[7]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring numbe
0	1	5	2	4	1	6	1	0	1	4	...	2	7	7	0	2	
1	0	5	2	9	1	0	1	0	0	4	...	2	7	7	0	2	
2	0	0	2	8	1	3	1	0	0	5	...	2	7	7	0	2	
3	1	5	3	8	1	6	1	0	1	5	...	2	7	7	0	2	
4	0	5	2	3	0	5	1	1	0	4	...	2	7	7	0	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
8119	0	3	2	4	0	5	0	0	0	11	...	2	5	5	0	1	
8120	0	5	2	4	0	5	0	0	0	11	...	2	5	5	0	0	
8121	0	2	2	4	0	5	0	0	0	5	...	2	5	5	0	1	
8122	1	3	3	4	0	8	1	0	1	0	...	1	7	7	0	2	
8123	0	5	2	4	0	5	0	0	0	11	...	2	5	5	0	1	

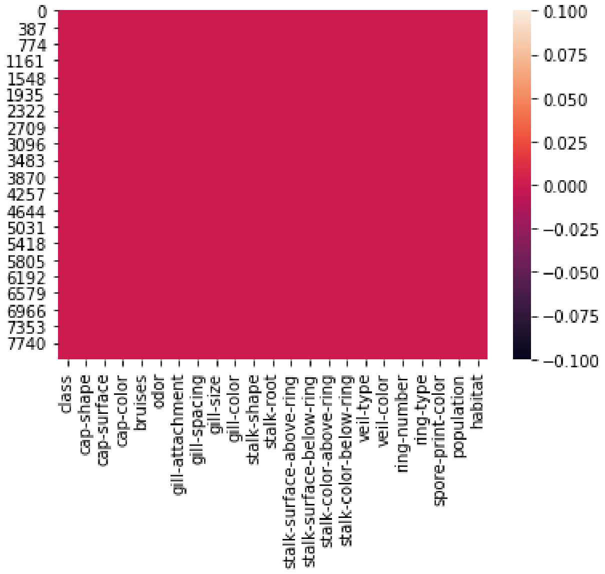
8124 rows × 23 columns

In [8]:

```
# identifing null values
sns.heatmap(mushroom.isnull())
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6a0167248>
```



In [9]:

```
mushroom.isnull().sum()
```

Out[9]:

```
class                       0
cap-shape                   0
cap-surface                 0
cap-color                   0
bruises                     0
odor                        0
gill-attachment             0
gill-spacing                0
gill-size                   0
gill-color                  0
stalk-shape                 0
stalk-root                  0
stalk-surface-above-ring    0
stalk-surface-below-ring    0
stalk-color-above-ring      0
stalk-color-below-ring      0
veil-type                   0
veil-color                  0
ring-number                 0
ring-type                   0
spore-print-color           0
population                  0
habitat                     0
dtype: int64
```

In [10]:

```
mushroom.skew()
```

Out[10]:

```
class                        0.071946
cap-shape                   -0.247052
```

```
cap-surface                -0.590859
cap-color                   0.706965
bruises                     0.342750
odor                       -0.080790
gill-attachment            -5.977076
gill-spacing                1.840088
gill-size                   0.825797
gill-color                  0.061410
stalk-shape                -0.271345
stalk-root                  0.947852
stalk-surface-above-ring   -1.098739
stalk-surface-below-ring   -0.757703
stalk-color-above-ring     -1.835434
stalk-color-below-ring     -1.791593
veil-type                   0.000000
veil-color                 -6.946944
ring-number                 2.701657
ring-type                  -0.290018
spore-print-color           0.548426
population                 -1.413096
habitat                     0.985548
dtype: float64
```

In [11]:

```
mushroom.corr()
```

Out[11]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	sta col abo ri
class	1.000000	0.052951	0.178446	-0.031384	-0.501530	-0.093552	0.129200	-0.348387	0.540024	-0.530566	...	-0.298801	0.1540
cap-shape	0.052951	1.000000	-0.050454	-0.048203	-0.035374	-0.021935	0.078865	0.013196	0.054050	-0.006039	...	-0.032591	0.0316
cap-surface	0.178446	-0.050454	1.000000	-0.019402	0.070228	0.045233	-0.034180	-0.282306	0.208100	-0.161017	...	0.107965	0.0660
cap-color	-0.031384	-0.048203	-0.019402	1.000000	-0.000764	-0.387121	0.041436	0.144259	-0.169464	0.084659	...	-0.047710	0.0023
bruises	-0.501530	-0.035374	0.070228	-0.000764	1.000000	-0.061825	0.137359	-0.299473	-0.369596	0.527120	...	0.458983	0.0835
odor	-0.093552	-0.021935	0.045233	-0.387121	-0.061825	1.000000	-0.059590	0.063936	0.310495	-0.129213	...	0.061820	0.1745
gill-attachment	0.129200	0.078865	-0.034180	0.041436	0.137359	-0.059590	1.000000	0.071489	0.108984	-0.128567	...	-0.116177	0.0992
gill-spacing	-0.348387	0.013196	-0.282306	0.144259	-0.299473	0.063936	0.071489	1.000000	-0.108333	0.100193	...	-0.213775	0.2745
gill-size	0.540024	0.054050	0.208100	-0.169464	-0.369596	0.310495	0.108984	-0.108333	1.000000	-0.516736	...	0.010894	0.2965
gill-color	-0.530566	-0.006039	-0.161017	0.084659	0.527120	-0.129213	-0.128567	0.100193	-0.516736	1.000000	...	0.257224	0.0582
stalk-shape	-0.102019	0.063794	-0.014123	-0.456496	0.099364	0.459766	0.186485	0.080895	0.214576	-0.175699	...	-0.034399	0.2234
stalk-root	-0.379361	0.030191	-0.126245	0.321274	0.244188	-0.205215	0.144063	0.350548	-0.344345	0.315080	...	0.087454	0.1571
stalk-surface-above-ring	-0.334593	-0.030417	0.089090	-0.060837	0.460824	0.118617	-0.088916	-0.212359	0.056310	0.224287	...	0.437164	0.1327
stalk-surface-below-ring	-0.298801	-0.032591	0.107965	-0.047710	0.458983	0.061820	-0.116177	-0.213775	0.010894	0.257224	...	1.000000	0.1069
stalk-color-above-ring	-0.154003	-0.031659	0.066050	0.002364	0.083538	0.174532	0.099299	0.274574	0.296548	-0.058299	...	0.106933	1.0000
stalk-color-below-ring	-0.146730	-0.030390	0.068885	0.008057	0.092874	0.169407	0.097160	0.253505	0.278708	-0.074781	...	0.110656	0.4915
veil-type	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	N

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	sta col abo ri
veil-color	0.145142	0.072560	-0.016603	0.036130	0.119770	-0.057747	0.897518	0.073363	0.103809	-0.097583	...	-[illegible]	0.0673
ring-number	-0.214366	-0.106534	-0.026147	0.005822	0.056788	0.111905	0.093236	0.243014	-0.171362	0.096054	...	0.040006	0.0849
ring-type	-0.411771	-0.025457	-0.106407	0.162513	0.692973	-0.281387	-0.146689	-0.195897	-0.460872	0.629398	...	0.394644	0.0488
spore-print-color	0.171961	-0.073416	0.230364	-0.293523	-0.285008	0.469055	-0.029524	0.047323	0.622991	-0.416135	...	0.130974	0.2715
population	0.298686	0.063413	0.021555	-0.144770	0.088137	-0.043623	0.165575	-0.529253	0.147682	-0.034090	...	0.046797	0.2402
habitat	0.217179	-0.042221	0.163887	0.033925	-0.075095	-0.026610	-0.030304	-0.154680	0.161418	-0.202972	...	-0.039628	0.0425

23 rows × 23 columns

In [12]:

```
for col in mushroom.columns:
    if mushroom.skew().loc[col]>0.55:
        mushroom[col]=np.log1p(mushroom[col])
```

In [13]:

```
#reduced skewness
mushroom.skew()
```

Out[13]:

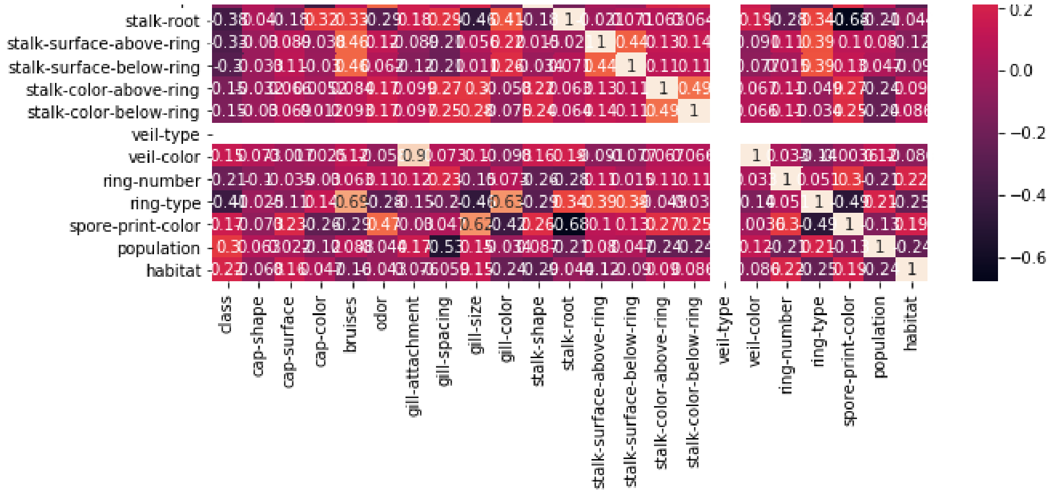
```
class                       0.071946
cap-shape                  -0.247052
cap-surface                -0.590859
cap-color                  -0.365280
bruises                     0.342750
odor                       -0.080790
gill-attachment            -5.977076
gill-spacing                1.840088
gill-size                   0.825797
gill-color                  0.061410
stalk-shape                -0.271345
stalk-root                  0.129453
stalk-surface-above-ring   -1.098739
stalk-surface-below-ring   -0.757703
stalk-color-above-ring     -1.835434
stalk-color-below-ring     -1.791593
veil-type                   0.000000
veil-color                 -6.946944
ring-number                 1.481287
ring-type                  -0.290018
spore-print-color           0.548426
population                 -1.413096
habitat                     0.342186
dtype: float64
```

In [14]:

```
plt.figure(figsize=(10,6))
sns.heatmap(mushroom.corr(),annot=True)
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6a09bf4c8>
```



In [15]:

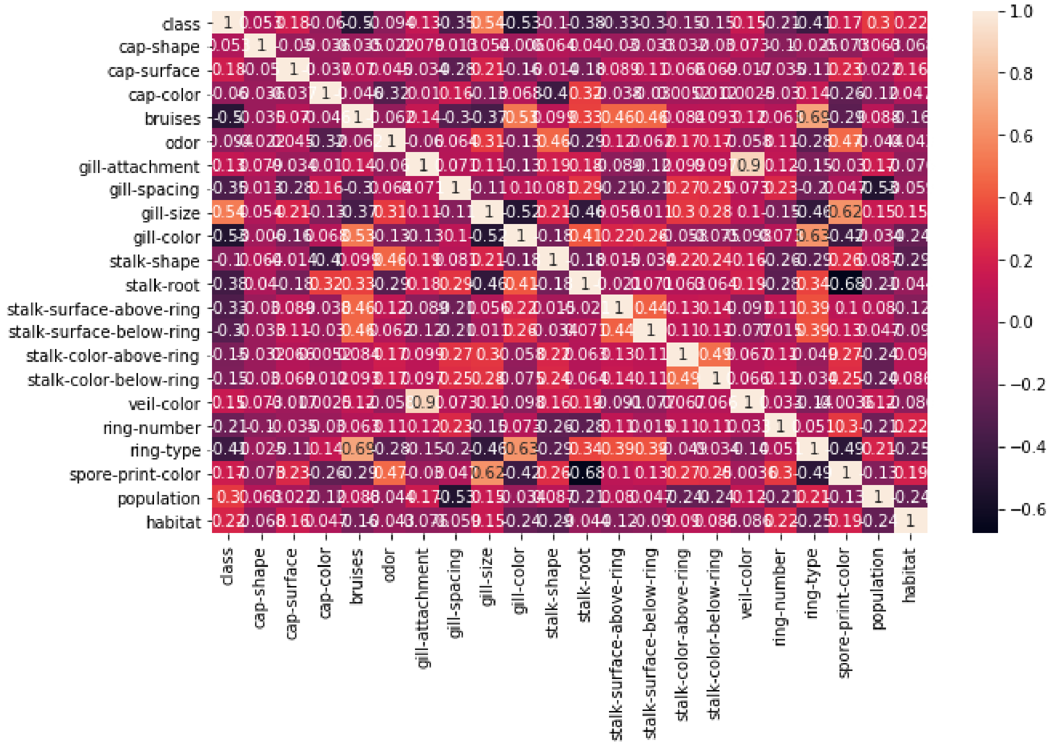
```
mushroom=mushroom.drop(['veil-type'],axis=1)
```

In [16]:

```
plt.figure(figsize=(10,6))
sns.heatmap(mushroom.corr(),annot=True)
```

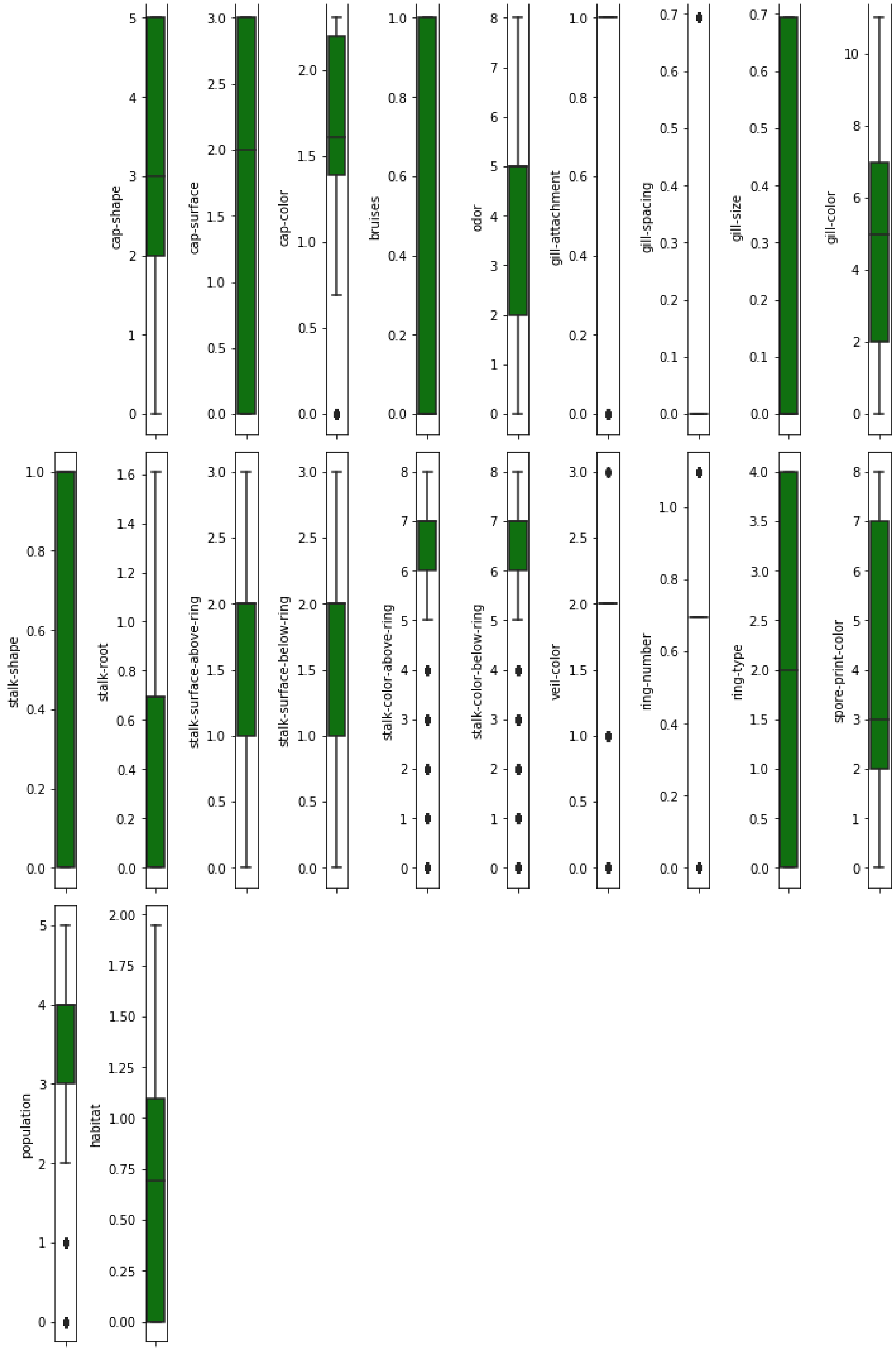
Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6a1891788>
```



In [17]:

```
col=mushroom.columns.values
ncol=10
nrow=10
plt.figure(figsize=(ncol,5*ncol))
for i in range(1,len(col)):
    plt.subplot(nrow,ncol,i+1)
    sns.boxplot(mushroom[col[i]],color='green',orient='v')
    plt.tight_layout()
```



In [18]:

```
#Removing outliers
from scipy.stats import zscore
z_score=abs(zscore(mushroom))
print(mushroom.shape)
mush=mushroom.loc[(z_score<3).all(axis=1)]
print(mush.shape)
```

```
(8124, 22)
(6472, 22)
```

In [19]:

```
mush
```

Out[19]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring
0	1	5	2	1.609438	1	6	1	0.000000	0.693147	4	...	2	2	7	7
1	0	5	2	2.302585	1	0	1	0.000000	0.000000	4	...	2	2	7	7
2	0	0	2	2.197225	1	3	1	0.000000	0.000000	5	...	2	2	7	7
3	1	5	3	2.197225	1	6	1	0.000000	0.693147	5	...	2	2	7	7
4	0	5	2	1.386294	0	5	1	0.693147	0.000000	4	...	2	2	7	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8113	1	3	3	1.098612	0	8	1	0.000000	0.693147	0	...	1	1	6	6
8116	1	3	3	1.609438	0	7	1	0.000000	0.693147	0	...	2	1	6	7
8117	1	3	2	1.098612	0	8	1	0.000000	0.693147	0	...	1	2	6	7
8118	1	3	3	1.609438	0	2	1	0.000000	0.693147	0	...	1	2	6	7
8122	1	3	3	1.609438	0	8	1	0.000000	0.693147	0	...	2	1	7	7

6472 rows × 22 columns

In [20]:

```
#x and y values allocation for training and testing
x=mush.iloc[:,1:-1]
```

In [21]:

```
x
```

Out[21]:

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	stalk-root	stalk-surface-above-ring	stalk-surface-below-ring	stalk-color-above-ring
0	5	2	1.609438	1	6	1	0.000000	0.693147	4	0	1.386294	2	2	7
1	5	2	2.302585	1	0	1	0.000000	0.000000	4	0	1.098612	2	2	7
2	0	2	2.197225	1	3	1	0.000000	0.000000	5	0	1.098612	2	2	7
3	5	3	2.197225	1	6	1	0.000000	0.693147	5	0	1.386294	2	2	7
4	5	2	1.386294	0	5	1	0.693147	0.000000	4	1	1.386294	2	2	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8113	3	3	1.098612	0	8	1	0.000000	0.693147	0	1	0.000000	1	1	6
8116	3	3	1.609438	0	7	1	0.000000	0.693147	0	1	0.000000	2	1	6
8117	3	2	1.098612	0	8	1	0.000000	0.693147	0	1	0.000000	1	2	6
8118	3	3	1.609438	0	2	1	0.000000	0.693147	0	1	0.000000	1	2	6
8122	3	3	1.609438	0	8	1	0.000000	0.693147	0	1	0.000000	2	1	7

6472 rows × 20 columns

In [22]:

```
x.shape
```

Out[22]:

```
(6472, 20)
```

In [23]:

```
y=mush.iloc[:,0]
```

In [24]:

```
y
```

Out[24]:

```
0       1
1       0
2       0
3       1
4       0
       ..
8113    1
8116    1
8117    1
8118    1
8122    1
Name: class, Length: 6472, dtype: int32
```

In [25]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=42)
```

In [26]:

```
#we are using follwing models for prediction
#LogisticRegression
#KNeighborsClassifier
#GaussianNB
#SVC
#DecisionTreeClassifier
#RandomForestClassifier
#AdaBoostClassifier
```

In [27]:

```
lr=LogisticRegression()
lr.fit(x_train,y_train)
lr.score(x_train,y_train)
pred=lr.predict(x_test)
print(accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
0.955200823892894
[[990  45]
 [ 42 865]]
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      1035
           1       0.95      0.95      0.95       907

    accuracy                           0.96      1942
   macro avg       0.95      0.96      0.96      1942
weighted avg       0.96      0.96      0.96      1942
```

In [28]:

```
lrscores=cross_val_score(lr,x,y,cv=5)
print(lrscores)
print(lrscores.mean(),lrscores.std())
```

```
[0.77606178 0.91351351 0.96136012 1.         0.93508501]
0.9172040841901739 0.07623571128762385
```

In [29]:

```
svc=SVC(kernel='rbf')
svc.fit(x_train,y_train)
svc.score(x_train,y_train)
predsvc=svc.predict(x_test)
print(accuracy_score(y_test,predsvc))
print(confusion_matrix(y_test,predsvc))
print(classification_report(y_test,predsvc))
```

```
0.9907312049433573
[[1034    1]
 [  17  890]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1035
           1       1.00      0.98      0.99       907

    accuracy                           0.99      1942
   macro avg       0.99      0.99      0.99      1942
weighted avg       0.99      0.99      0.99      1942
```

In [30]:

```
svcscores=cross_val_score(svc,x,y,cv=5)
print(svcscores)
print(svcscores.mean(),svcscores.std())
```

```
[0.73050193 0.95057915 0.96445131 1.         0.92581144]
0.9142687664480554 0.09496321361749795
```

In [31]:

```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
knn.score(x_train,y_train)
predknn=knn.predict(x_test)
print(accuracy_score(y_test,predknn))
print(confusion_matrix(y_test,predknn))
print(classification_report(y_test,predknn))
```

```
0.9984552008238929
[[1033    2]
 [   1  906]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1035
           1       1.00      1.00      1.00       907

    accuracy                           1.00      1942
   macro avg       1.00      1.00      1.00      1942
weighted avg       1.00      1.00      1.00      1942
```

In [32]:

```
knnscores=cross_val_score(knn,x,y,cv=5)
print(knnscores)
print(knnscores.mean(),knnscores.std())
```

```
[0.76833977 0.98841699 0.9992272  1.         0.96290572]
0.9437779355862818 0.088738502778172
```

In [33]:

```
gnb=GaussianNB()
```

```
gnb=GaussianNB()
gnb.fit(x_train,y_train)
gnb.score(x_train,y_train)
predgnb=gnb.predict(x_test)
print(accuracy_score(y_test,predgnb))
print(confusion_matrix(y_test,predgnb))
print(classification_report(y_test,predgnb))
```

```
0.8558187435633368
[[985  50]
 [230 677]]
              precision    recall  f1-score   support

           0       0.81      0.95      0.88      1035
           1       0.93      0.75      0.83       907

    accuracy                           0.86      1942
   macro avg       0.87      0.85      0.85      1942
weighted avg       0.87      0.86      0.85      1942
```

In [34]:

```
gnbscores=cross_val_score(gnb,x,y,cv=5)
print(gnbscores)
print(gnbscores.mean(),gnbscores.std())
```

```
[0.65559846 0.64633205 0.92194745 1.         0.92581144]
0.8299378778204126 0.14878438201274055
```

In [35]:

```
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
dtc.score(x_train,y_train)
preddtc=dtc.predict(x_test)
print(accuracy_score(y_test,preddtc))
print(confusion_matrix(y_test,preddtc))
print(classification_report(y_test,preddtc))
```

```
1.0
[[1035    0]
 [   0  907]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1035
           1       1.00      1.00      1.00       907

    accuracy                           1.00      1942
   macro avg       1.00      1.00      1.00      1942
weighted avg       1.00      1.00      1.00      1942
```

In [36]:

```
dtcscores=cross_val_score(dtc,x,y,cv=5)
print(dtcscores)
print(dtcscores.mean(),dtcscores.std())
```

```
[0.9011583  1.         1.         1.         0.94435858]
0.9691033758421703 0.0402309391998156
```

In [37]:

```
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
1.0
[[1035    0]
 [   0  907]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1035
           1       1.00      1.00      1.00       907

    accuracy                           1.00      1942
   macro avg       1.00      1.00      1.00      1942
weighted avg       1.00      1.00      1.00      1942
```

In [38]:

```
rfscores=cross_val_score(rf,x,y,cv=5)
print(rfscores)
print(rfscores.mean(),rfscores.std())
```

```
[0.8023166  1.         1.         1.         0.92581144]
0.9456256079440004 0.07720077199420258
```

In [39]:

```
#DecisionTreeClassifier and randomForestClassifier are best models among all models
import joblib
joblib.dump(dtc,'mushroom.pkl')
```

Out[39]:

```
['mushroom.pkl']
```

In [ ]: