In [30]:

```
import pandas as pd
import re
import string
import scipy
import numpy as np
import seaborn as sns
from sklearn import preprocessing
from sklearn.feature_extraction.text import *
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score

from astropy.table import Table, Column
import matplotlib.pyplot as plt
```

In [31]:

```
train_data = pd.read_csv("train.csv")
print("Train Dataset:\n")
print(train_data)
```

```
Train Dataset:

    cap-surface cap-color     odor gill-spacing gill-size gill-color  \
0        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
1        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
2        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
3        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
4        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
5        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
6        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
7        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
8        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
9        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
10       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
11       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
12       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
13       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
14       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      BLACK
15       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      BLACK
```

In [32]:

```
train_data.dtypes
```

Out[32]:

```
cap-surface          object
cap-color            object
odor                 object
gill-spacing         object
gill-size            object
gill-color           object
ring-number          object
spore-print-color    object
habitat              object
mushroom             object
dtype: object
```

In [5]:

```
test_data = pd.read_csv("test.csv")
print("Test Dataset:\n")
print(test_data)
```

```
Test Dataset:

   cap-surface cap-color     odor gill-spacing gill-size gill-color  \
0        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
1        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
2        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
3        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
4        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
5       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
6       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
7       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
8       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
9       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
10      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
11      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
12      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
13      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
14      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
15      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
```

In [6]:

```
test_data.dtypes
```

Out[6]:

```
cap-surface          object
cap-color            object
odor                 object
gill-spacing         object
gill-size            object
gill-color           object
ring-number          object
spore-print-color    object
habitat              object
mushroom             object
dtype: object
```

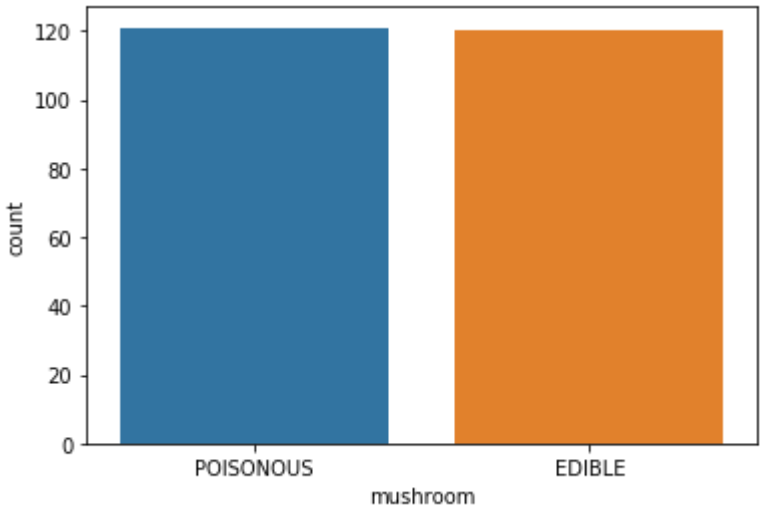
In [7]:

```
print("Total n.o of POISONOUS and EDIBLE in test data set")
sns.countplot("mushroom",data=train_data)
```

```
Total n.o of POISONOUS and EDIBLE in test data set
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c122345ac8>
```



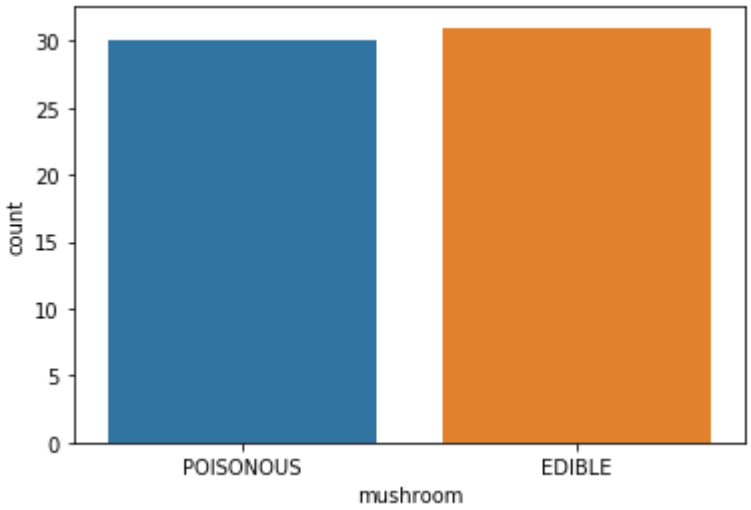
In [8]:

```
print("Total n.o of POISONOUS and EDIBLE in test data set")
sns.countplot("mushroom",data=test_data)
```

```
Total n.o of POISONOUS and EDIBLE in test data set
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1225fc550>
```



In [9]:

```
le_cap_s = preprocessing.LabelEncoder()
le_cap_c = preprocessing.LabelEncoder()
le_odor = preprocessing.LabelEncoder()
le_gill_sp = preprocessing.LabelEncoder()
le_gill_si = preprocessing.LabelEncoder()
le_gill_c = preprocessing.LabelEncoder()
le_ring_n = preprocessing.LabelEncoder()
le_spore_p = preprocessing.LabelEncoder()
le_habitat = preprocessing.LabelEncoder()
le_mushroom = preprocessing.LabelEncoder()
```

In [10]:

```
train_data['encoded_mushroom'] = le_mushroom.fit_transform(train_data['mushroom'])

print("Gender attribute encoding in train dataset  :\n")
print(train_data[["mushroom","encoded_mushroom"]])
```

```
Gender attribute encoding in train dataset  :

      mushroom  encoded_mushroom
0    POISONOUS                 1
1    POISONOUS                 1
2    POISONOUS                 1
3    POISONOUS                 1
4    POISONOUS                 1
5    POISONOUS                 1
6    POISONOUS                 1
7    POISONOUS                 1
8    POISONOUS                 1
9    POISONOUS                 1
10   POISONOUS                 1
11   POISONOUS                 1
12   POISONOUS                 1
13   POISONOUS                 1
14   POISONOUS                 1
15   POISONOUS                 1
16   POISONOUS                 1
17   POISONOUS                 1
18   POISONOUS                 1
19   POISONOUS                 1
20   POISONOUS                 1
21   POISONOUS                 1
22   POISONOUS                 1
23   POISONOUS                 1
24   POISONOUS                 1
25   POISONOUS                 1
26   POISONOUS                 1
27   POISONOUS                 1
28   POISONOUS                 1
29   POISONOUS                 1
..         ...               ...
211     EDIBLE                 0
212     EDIBLE                 0
213     EDIBLE                 0
214     EDIBLE                 0
215     EDIBLE                 0
216     EDIBLE                 0
217     EDIBLE                 0
218     EDIBLE                 0
219     EDIBLE                 0
220     EDIBLE                 0
221     EDIBLE                 0
222     EDIBLE                 0
223     EDIBLE                 0
224     EDIBLE                 0
225     EDIBLE                 0
226     EDIBLE                 0
227     EDIBLE                 0
228     EDIBLE                 0
229     EDIBLE                 0
```

```
230     EDIBLE                0
231     EDIBLE                0
232     EDIBLE                0
233     EDIBLE                0
234     EDIBLE                0
235     EDIBLE                0
236     EDIBLE                0
237     EDIBLE                0
238     EDIBLE                0
239     EDIBLE                0
240     EDIBLE                0

[241 rows x 2 columns]
```

In [11]:

```
train_data_pre = pd.read_csv("train.csv")
```

In [12]:

```
print("train dataset without encoding :\n")
print(train_data_pre,"\n")
train_data_en = pd.read_csv("train.csv")
train_data_en['cap-surface'] = le_cap_s.fit_transform(train_data_en['cap-surface'])
train_data_en['cap-color'] = le_cap_c.fit_transform(train_data_en['cap-color'])
train_data_en['odor'] = le_odor.fit_transform(train_data_en['odor'])
train_data_en['gill-spacing'] = le_gill_sp.fit_transform(train_data_en['gill-spacing'])
train_data_en['gill-size'] = le_gill_si.fit_transform(train_data_en['gill-size'])
train_data_en['gill-color'] = le_gill_c.fit_transform(train_data_en['gill-color'])
train_data_en['ring-number'] = le_ring_n.fit_transform(train_data_en['ring-number'])
train_data_en['spore-print-color'] = le_spore_p.fit_transform(train_data_en['spore-print-co
train_data_en['habitat'] = le_habitat.fit_transform(train_data_en['habitat'])
train_data_en['mushroom'] = le_mushroom.fit_transform(train_data_en['mushroom'])

print("train dataset with encoding :\n")
print(train_data_en)
```

```
train dataset without encoding :

    cap-surface cap-color     odor gill-spacing gill-size gill-color  \
0        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
1        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
2        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
3        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
4        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
5        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
6        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
7        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
8        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
9        SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
10       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
11       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
12       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
13       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
14       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      BLACK
15       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      BLACK
```

In [13]:

```
test_data_woe = pd.read_csv("test.csv")
print("test dataset without encoding :\n")
print(test_data_woe,"\n")
test_data['cap-surface'] = le_cap_s.fit_transform(test_data['cap-surface'])
test_data['cap-color'] = le_cap_c.fit_transform(test_data['cap-color'])
test_data['odor'] = le_odor.fit_transform(test_data['odor'])
test_data['gill-spacing'] = le_gill_sp.fit_transform(test_data['gill-spacing'])
test_data['gill-size'] = le_gill_si.fit_transform(test_data['gill-size'])
test_data['gill-color'] = le_gill_c.fit_transform(test_data['gill-color'])
test_data['ring-number'] = le_ring_n.fit_transform(test_data['ring-number'])
test_data['spore-print-color'] = le_spore_p.fit_transform(test_data['spore-print-color'])
test_data['habitat'] = le_habitat.fit_transform(test_data['habitat'])
test_data['mushroom'] = le_mushroom.fit_transform(test_data['mushroom'])

print("test dataset with encoding :\n")
print(test_data)
```

```
test dataset without encoding :

   cap-surface cap-color     odor gill-spacing gill-size gill-color  \
0        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
1        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
2        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
3        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
4        SCALY     BROWN  PUNGENT        CLOSE    NARROW      BROWN
5       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
6       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
7       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
8       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
9       SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
10      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
11      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
12      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW      WHITE
13      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
14      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
15      SMOOTH     WHITE  PUNGENT        CLOSE    NARROW       PINK
```

In [14]:

```
X_train = train_data_en[['cap-surface','cap-color','odor','gill-spacing','gill-size','gill-
Y_train = train_data_en[['mushroom']]

X_test = test_data[['cap-surface','cap-color','odor','gill-spacing','gill-size','gill-color
Y_test = test_data[['mushroom']]
```

In [15]:

```
lr = LogisticRegression()

pred = lr.fit(X_train, Y_train).predict(X_test)

print("LogisticRegression accuracy : ",accuracy_score(Y_test, pred, normalize = True))
```

```
LogisticRegression accuracy :  1.0

C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\sklearn\linear_model\logis
tic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:724: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples, ), for example usin
g ravel().
  y = column_or_1d(y, warn=True)
```

In [16]:

```
ls = LinearSVC()

pred_ls = ls.fit(X_train, Y_train).predict(X_test)


print("LinearSVC accuracy : ",accuracy_score(Y_test, pred_ls))
```

```
LinearSVC accuracy :  1.0

C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:724: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples, ), for example usin
g ravel().
  y = column_or_1d(y, warn=True)
```

In [17]:

```
rfc = RandomForestClassifier()

pred_rfc = rfc.fit(X_train, Y_train).predict(X_test)

print("RandomForestClassifier accuracy : ",accuracy_score(Y_test, pred_rfc))
```

```
RandomForestClassifier accuracy :  1.0

C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\sklearn\ensemble\forest.p
y:245: FutureWarning: The default value of n_estimators will change from 10
in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: D
ataConversionWarning: A column-vector y was passed when a 1d array was expec
ted. Please change the shape of y to (n_samples,), for example using ravel
().
  This is separate from the ipykernel package so we can avoid doing imports
until
```

In [18]:

```
ber = BernoulliNB()

pred_ber = ber.fit(X_train, Y_train).predict(X_test)

print("BernoulliNB accuracy : ",accuracy_score(Y_test, pred_ber))
```

```
BernoulliNB accuracy :  1.0

C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:724: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples, ), for example usin
g ravel().
  y = column_or_1d(y, warn=True)
```

In [19]:

```
print("Detailed performance of all the models")
print("=====================================+")
print("+-----------------------------------+")
print("|        Model       |    Accuracy   |")
print("+-----------------------------------+")
print("| LogisticRegression |     ",accuracy_score(Y_test, pred, normalize = True),"     |"
print("| RandFrstClassifier |     ",accuracy_score(Y_test, pred_rfc),"     |")
print("|     LinearSVC      |     ",accuracy_score(Y_test, pred_ls),"     |")
print("|    BernoulliNB     |     ",accuracy_score(Y_test, pred_ber),"     |")
print("+-----------------------------------+")
print("")
print("                Best Model           ")
print("=====================================+")
print("+-----------------------------------+")
print("|        Model       |    Accuracy   |")
print("+-----------------------------------+")
print("| RandFrstClassifier |     ",accuracy_score(Y_test, pred_rfc),"     |")
print("+-----------------------------------+")
```

```
Detailed performance of all the models
=====================================+
+-----------------------------------+
|        Model       |    Accuracy   |
+-----------------------------------+
| LogisticRegression |      1.0      |
| RandFrstClassifier |      1.0      |
|     LinearSVC      |      1.0      |
|    BernoulliNB     |      1.0      |
+-----------------------------------+

                Best Model
=====================================+
+-----------------------------------+
|        Model       |    Accuracy   |
+-----------------------------------+
| RandFrstClassifier |      1.0      |
+-----------------------------------+
```

In [20]:

```
print(train_data_en)
```

```
     cap-surface  cap-color  odor  gill-spacing  gill-size  gill-color  \
0              1          2     1             0          1           4
1              1          2     1             0          1           4
2              1          2     1             0          1           4
3              1          2     1             0          1           4
4              1          2     1             0          1           4
5              1          2     1             0          1           4
6              1          2     1             0          1           3
7              1          2     1             0          1           3
8              1          2     1             0          1           3
9              1          2     1             0          1           3
10             1          2     1             0          1           3
11             1          2     1             0          1           3
12             1          2     1             0          1           3
13             1          2     1             0          1           3
14             1          2     1             0          1           0
15             1          2     1             0          1           0
16             1          2     1             0          1           0
17             1          2     1             0          1           0
```

In [21]:

```
print("test dataset with encoding:")
print(test_data)
```

```
test dataset with encoding:
    cap-surface  cap-color  odor  gill-spacing  gill-size  gill-color  \
0             0          0     1             0          1           1
1             0          0     1             0          1           1
2             0          0     1             0          1           1
3             0          0     1             0          1           1
4             0          0     1             0          1           1
5             1          2     1             0          1           4
6             1          2     1             0          1           4
7             1          2     1             0          1           4
8             1          2     1             0          1           4
9             1          2     1             0          1           4
10            1          2     1             0          1           4
11            1          2     1             0          1           4
12            1          2     1             0          1           4
13            1          2     1             0          1           3
14            1          2     1             0          1           3
15            1          2     1             0          1           3
16            1          2     1             0          1           3
```

In [22]:

```
combined_data = train_data_en.append(test_data)
print("Combinded Data (Train + Test) :")
print(combined_data)
```

```
Combinded Data (Train + Test) :
    cap-surface  cap-color  odor  gill-spacing  gill-size  gill-color  \
0             1          2     1             0          1           4
1             1          2     1             0          1           4
2             1          2     1             0          1           4
3             1          2     1             0          1           4
4             1          2     1             0          1           4
5             1          2     1             0          1           4
6             1          2     1             0          1           3
7             1          2     1             0          1           3
8             1          2     1             0          1           3
9             1          2     1             0          1           3
10            1          2     1             0          1           3
11            1          2     1             0          1           3
12            1          2     1             0          1           3
13            1          2     1             0          1           3
14            1          2     1             0          1           0
15            1          2     1             0          1           0
16            1          2     1             0          1           0
```

In [23]:

```
X_train = combined_data[['cap-surface','cap-color','odor','gill-spacing','gill-size','gill-
Y_train = combined_data[['mushroom']]
```

In [24]:

```
rfc = RandomForestClassifier()

pred_rfc = rfc.fit(X_train, Y_train)
```

```
C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\sklearn\ensemble\forest.p
y:245: FutureWarning: The default value of n_estimators will change from 10
in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\Users\Sauda Maryam\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: D
ataConversionWarning: A column-vector y was passed when a 1d array was expec
ted. Please change the shape of y to (n_samples,), for example using ravel
().
  This is separate from the ipykernel package so we can avoid doing imports
until
```

In [25]:

```
cap_surface = input ("Please enter cap-surface (SCALY/SMOOTH) :")
cap_color = input ("Please enter cap-color (WHITE/BROWN/GRAY) :")
odor = input ("Please enter odor (PUNGENT/NONE) :")
gill_spacing = input ("Enter gill-spacing (CLOSE/CROWDED) :")
gill_size = input ("Enter gill-size (NARROW/BROAD) :")
gill_color = input ("Please enter gill-color (WHITE/PINK/BLACK/BROWN/CHOCOLATE) :")
ring_number = input ("Please enter ring-number (ONE) :")
spore_printt_color = input ("Please enter spore-printt-color (BROWN/BLACK) :")
habitat = input ("Enter habitat (GRASSES/URBAN) :")
```

```
Please enter cap-surface (SCALY/SMOOTH) :
Please enter cap-color (WHITE/BROWN/GRAY) :
Please enter odor (PUNGENT/NONE) :
Enter gill-spacing (CLOSE/CROWDED) :
Enter gill-size (NARROW/BROAD) :
Please enter gill-color (WHITE/PINK/BLACK/BROWN/CHOCOLATE) :
Please enter ring-number (ONE) :
Please enter spore-printt-color (BROWN/BLACK) :
Enter habitat (GRASSES/URBAN) :
```

In [26]:

```
user_data = {
    'cap-surface' : [cap_surface],
    'cap-color' : [cap_color],
    'odor'    : [odor],
    'gill-spacing'   : [gill_spacing],
    'gill-size'   : [gill_size],
    'gill-color' : [gill_color],
    'ring-number' : [ring_number],
    'spore-print-color'    : [spore_printt_color],
    'habitat'   : [habitat]
}

user_input = pd.DataFrame(user_data)
print("User input in actual DataFrame format:\n")
print(user_input)
```

```
User input in actual DataFrame format:

  cap-surface cap-color odor gill-spacing gill-size gill-color ring-number
\
0


  spore-print-color habitat
0
```

In [27]:

```
user_input['cap-surface'] = le_cap_s.fit_transform(user_input['cap-surface'])
user_input['cap-color'] = le_cap_c.fit_transform(user_input['cap-color'])
user_input['odor'] = le_odor.fit_transform(user_input['odor'])
user_input['gill-spacing'] = le_gill_sp.fit_transform(user_input['gill-spacing'])
user_input['gill-size'] = le_gill_si.fit_transform(user_input['gill-size'])
user_input['gill-color'] = le_gill_c.fit_transform(user_input['gill-color'])
user_input['ring-number'] = le_ring_n.fit_transform(user_input['ring-number'])
user_input['spore-print-color'] = le_spore_p.fit_transform(user_input['spore-print-color'])
user_input['habitat'] = le_habitat.fit_transform(user_input['habitat'])

print("User input in encoded DataFrame format : \n")
print(user_input)
```

```
User input in encoded DataFrame format :

   cap-surface  cap-color  odor  gill-spacing  gill-size  gill-color  \
0            0          0     0             0          0           0

   ring-number  spore-print-color  habitat
0            0                  0        0
```

In [28]:

```
df_woe = pd.DataFrame(user_data)
print("User input in actual DataFrame format : \n")
print(df_woe,"\n")
print("User input in encoded DataFrame format : \n")
print(user_input)
```

```
User input in actual DataFrame format :

  cap-surface cap-color odor gill-spacing gill-size gill-color ring-number
\
0

  spore-print-color habitat
0

User input in encoded DataFrame format :

   cap-surface  cap-color  odor  gill-spacing  gill-size  gill-color  \
0            0          0     0             0          0           0

   ring-number  spore-print-color  habitat
0            0                  0        0
```

In [33]:

```
result = pred_rfc.predict(user_input)
output = le_mushroom.inverse_transform(result)
print ("Prediction : ",output)
```

```
Prediction :  ['EDIBLE']
```

In [ ]:

```

```

In [ ]:

```

```