In [1]:

```
import pandas as pd
df= pd.read_csv('mushrooms.csv')
print (df)
```

```
     class cap-shape cap-surface cap-color bruises odor gill-attachment  \
0        p         x           s         n       t    p               f
1        e         x           s         y       t    a               f
2        e         b           s         w       t    l               f
3        p         x           y         w       t    p               f
4        e         x           s         g       f    n               f
...    ...       ...         ...       ...     ...  ...             ...
8119     e         k           s         n       f    n               a
8120     e         x           s         n       f    n               a
8121     e         f           s         n       f    n               a
8122     p         k           y         n       f    y               f
8123     e         x           s         n       f    n               a

     gill-spacing gill-size gill-color  ... stalk-surface-below-ring  \
0               c         n          k  ...                        s
1               c         b          k  ...                        s
2               c         b          n  ...                        s
3               c         n          n  ...                        s
4               w         b          k  ...                        s
...           ...       ...        ...  ...                      ...
8119            c         b          y  ...                        s
8120            c         b          y  ...                        s
8121            c         b          n  ...                        s
8122            c         n          b  ...                        k
8123            c         b          y  ...                        s

     stalk-color-above-ring stalk-color-below-ring veil-type veil-color  \
0                         w                       w         p          w
1                         w                       w         p          w
2                         w                       w         p          w
3                         w                       w         p          w
4                         w                       w         p          w
...                     ...                     ...       ...        ...
8119                      o                       o         p          o
8120                      o                       o         p          n
8121                      o                       o         p          o
8122                      w                       w         p          w
8123                      o                       o         p          o

     ring-number ring-type spore-print-color population habitat
0              o         p                 k          s       u
1              o         p                 n          n       g
2              o         p                 n          n       m
3              o         p                 k          s       u
4              o         e                 n          a       g
...          ...       ...               ...        ...     ...
8119           o         p                 b          c       l
8120           o         p                 b          v       l
8121           o         p                 b          c       l
8122           o         e                 w          v       l
8123           o         p                 o          c       l

[8124 rows x 23 columns]
```

In [2]:

```
df.shape
```

Out[2]:

```
(8124, 23)
```

In [3]:

```
df.head()
```

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-number	r t
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	

5 rows × 23 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   class                     8124 non-null   object
 1   cap-shape                 8124 non-null   object
 2   cap-surface               8124 non-null   object
 3   cap-color                 8124 non-null   object
 4   bruises                   8124 non-null   object
 5   odor                      8124 non-null   object
 6   gill-attachment           8124 non-null   object
 7   gill-spacing              8124 non-null   object
 8   gill-size                 8124 non-null   object
 9   gill-color                8124 non-null   object
 10  stalk-shape               8124 non-null   object
 11  stalk-root                8124 non-null   object
 12  stalk-surface-above-ring  8124 non-null   object
 13  stalk-surface-below-ring  8124 non-null   object
 14  stalk-color-above-ring    8124 non-null   object
 15  stalk-color-below-ring    8124 non-null   object
 16  veil-type                 8124 non-null   object
 17  veil-color                8124 non-null   object
 18  ring-number               8124 non-null   object
 19  ring-type                 8124 non-null   object
 20  spore-print-color         8124 non-null   object
 21  population                8124 non-null   object
 22  habitat                   8124 non-null   object
dtypes: object(23)
memory usage: 1.4+ MB
```

In [5]:

```
df.describe()
```

Out[5]:

	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

key observation-count shows highest categorical values unique shows unique values frequecy shows the frequency

In [6]:

```
df['class'].unique()
```

Out[6]:

```
array(['p', 'e'], dtype=object)
```

In [7]:

```
df['class'].value_counts()
```

Out[7]:

```
e    4208
p    3916
Name: class, dtype: int64
```

In [8]:

```
df = df.astype('category')
df.dtypes
```

Out[8]:

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

key observation-p,e are unique values all categorical data

ENCODING

In [31]:

```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
for column in df.columns:
    df[column] = labelencoder.fit_transform(df[column])
    df_new=df[column]
```

In [32]:

```
df.head()
```

Out[32]:

	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-number	r t
0	1	5	2	4	1	6	1	0	1	4	...	2	7	7	0	2	1	
1	0	5	2	9	1	0	1	0	0	4	...	2	7	7	0	2	1	
2	0	0	2	8	1	3	1	0	0	5	...	2	7	7	0	2	1	
3	1	5	3	8	1	6	1	0	1	5	...	2	7	7	0	2	1	
4	0	5	2	3	0	5	1	1	0	4	...	2	7	7	0	2	1	

5 rows × 23 columns

key observation-converted to numerical data

co-rellation of variables

In [26]:

```
import seaborn as sns
import matplotlib as plt
import numpy as np
corr_hmap=df.corr()
plt.figure(figsize=(8,7))
sns.heatmap(corr_hmap,annot=True)
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-843435b67cc9> in <module>
      3 import numpy as np
      4 corr_hmap=df.corr()
----> 5 plt.figure(figsize=(8,7))
      6 sns.heatmap(corr_hmap,annot=True)
      7 plt.show()

TypeError: 'module' object is not callable
```

splitting the data

In [35]:

```
x=df.iloc[:,0:-1]
x.head()
```

Out[35]:

	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	veil-type	veil-color	
0	1	5	2	4	1	6	1	0	1	4	...	2	2	7	7	0	2	
1	0	5	2	9	1	0	1	0	0	4	...	2	2	7	7	0	2	
2	0	0	2	8	1	3	1	0	0	5	...	2	2	7	7	0	2	
3	1	5	3	8	1	6	1	0	1	5	...	2	2	7	7	0	2	
4	0	5	2	3	0	5	1	1	0	4	...	2	2	7	7	0	2	

5 rows × 22 columns

In [37]:

```
y=df.iloc[:,-1]
y.head()
```

Out[37]:

```
0    5
1    1
2    3
3    5
4    1
Name: habitat, dtype: int32
```

In [39]:

```
import numpy as np
import pandas as pd
import sklearn
import seaborn as sns
from sklearn.model_selection import train_test_split
```

In [40]:

```
x.shape
```

Out[40]:

```
(8124, 22)
```

In [41]:

```
y.shape
```

Out[41]:

```
(8124,)
```

In [42]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

In [43]:

```
x_train.shape
```

Out[43]:

```
(5443, 22)
```

In [44]:

```
y_train.shape
```

Out[44]:

```
(5443,)
```

classification method

1.Decision Tree Classification

In [60]:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
print("Test Accuracy: {}%".format(round(dt.score(x_test, y_test)*100, 2)))
```

```
Test Accuracy: 49.42%
```

predicting and estimating the results

predicting and estimating the results

2.Logistic Regression

In [56]:

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
logistic_regression= LogisticRegression()
logistic_regression.fit(x_train,y_train)
y_pred=logistic_regression.predict(x_test)
print('Accuracy: ',metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy:  0.6482655725475569
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

In [58]:

```
from sklearn.neighbors import KNeighborsClassifier
best_Kvalue = 0
best_score = 0
for i in range(1,10):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    if knn.score(x_test, y_test) > best_score:
        best_score = knn.score(x_train, y_train)
        best_Kvalue = i
print("Best KNN Value: {}".format(best_Kvalue))
print("Test Accuracy: {}%".format(round(best_score*100,2)))
```

```
Best KNN Value: 1
Test Accuracy: 75.6%
```

In [ ]:

```

```

In [62]:

```
from sklearn.svm import SVC
svm = SVC(random_state=42, gamma="auto")
svm.fit(x_train, y_train)
print("Test Accuracy: {}%".format(round(svm.score(x_test, y_test)*100, 2)))
```

```
Test Accuracy: 62.14%
```

In [71]:

```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(svm, x, y, cv=5)
print("Cross validation score of svc model :", scr.mean())
```

```
Cross validation score of svc model : 0.47572459264873057
```

In [65]:

```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(dt, x, y, cv=5)
```

```
print("Cross validation score of DecisionTree model :", scr.mean())
```

```
Cross validation score of DecisionTree model : 0.4464230390299355
```

In [69]:

```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(knn, x, y, cv=5)
print("Cross validation score of KNeighbors model :", scr.mean())
```

```
Cross validation score of KNeighbors model : 0.46599583175445247
```

OBSERVATION-KNN is the best model