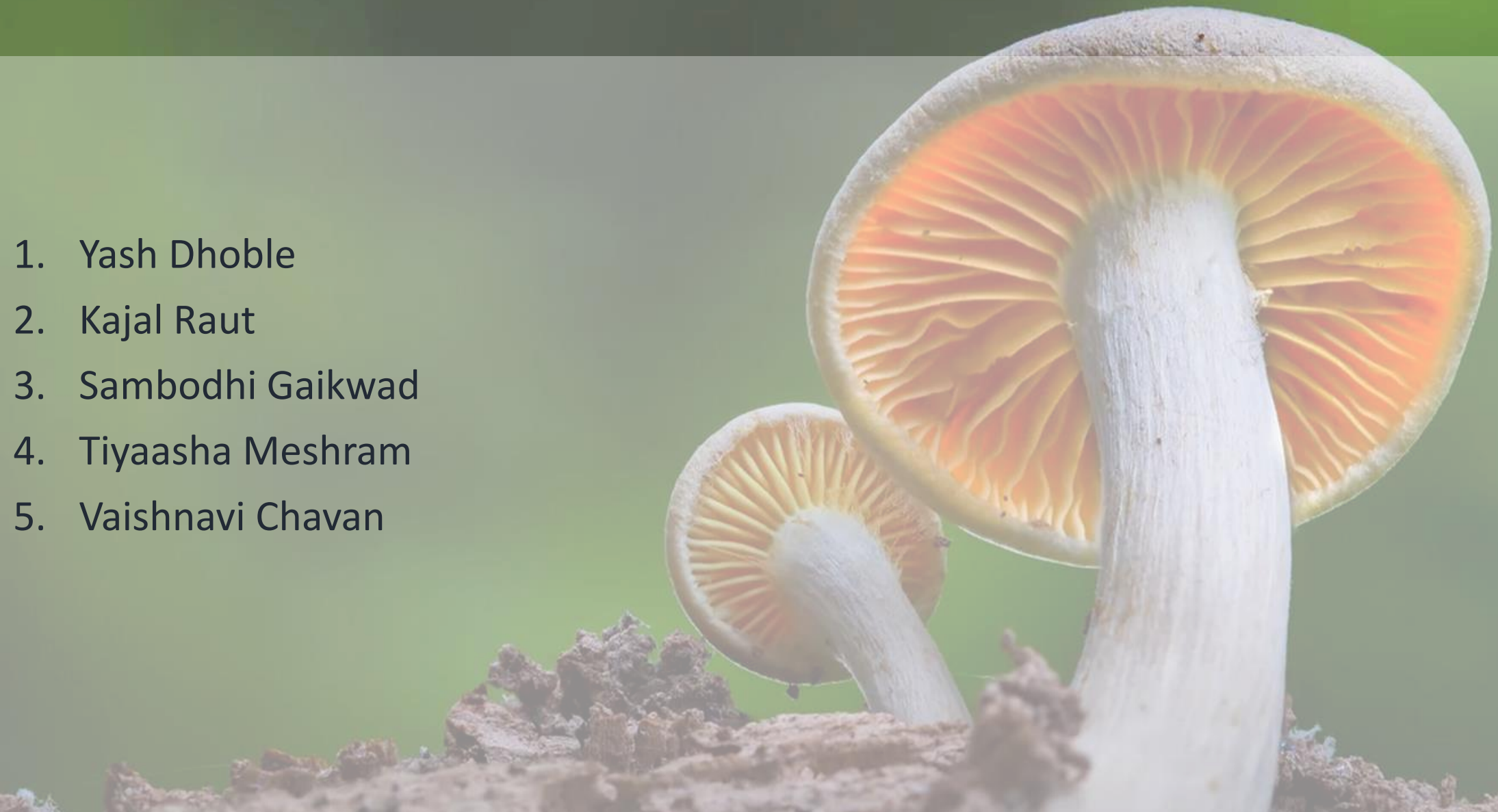
EDA of Mushroom Dataset

Meet the Team

1. Yash Dhoble
2. Kajal Raut
3. Sambodhi Gaikwad
4. Tiyaasha Meshram
5. Vaishnavi Chavan



Introduction to the Mushroom Dataset

- The mushroom dataset is a collection of information about various types of mushrooms and whether they are poisonous or edible.
- The dataset consists of 8124 observations and 23 variables, including the target variable "class", which indicates whether the mushroom is edible or poisonous.
- Data is purely categorical.

```
In [7]:  1  data.dtypes
         executed in 61ms, finished 14:58:56 2023-04-18

Out[7]:  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
```

Data Cleaning

First we checked for Null and Nan Values.

```
In [8]:  1 data.isnull().sum()
```

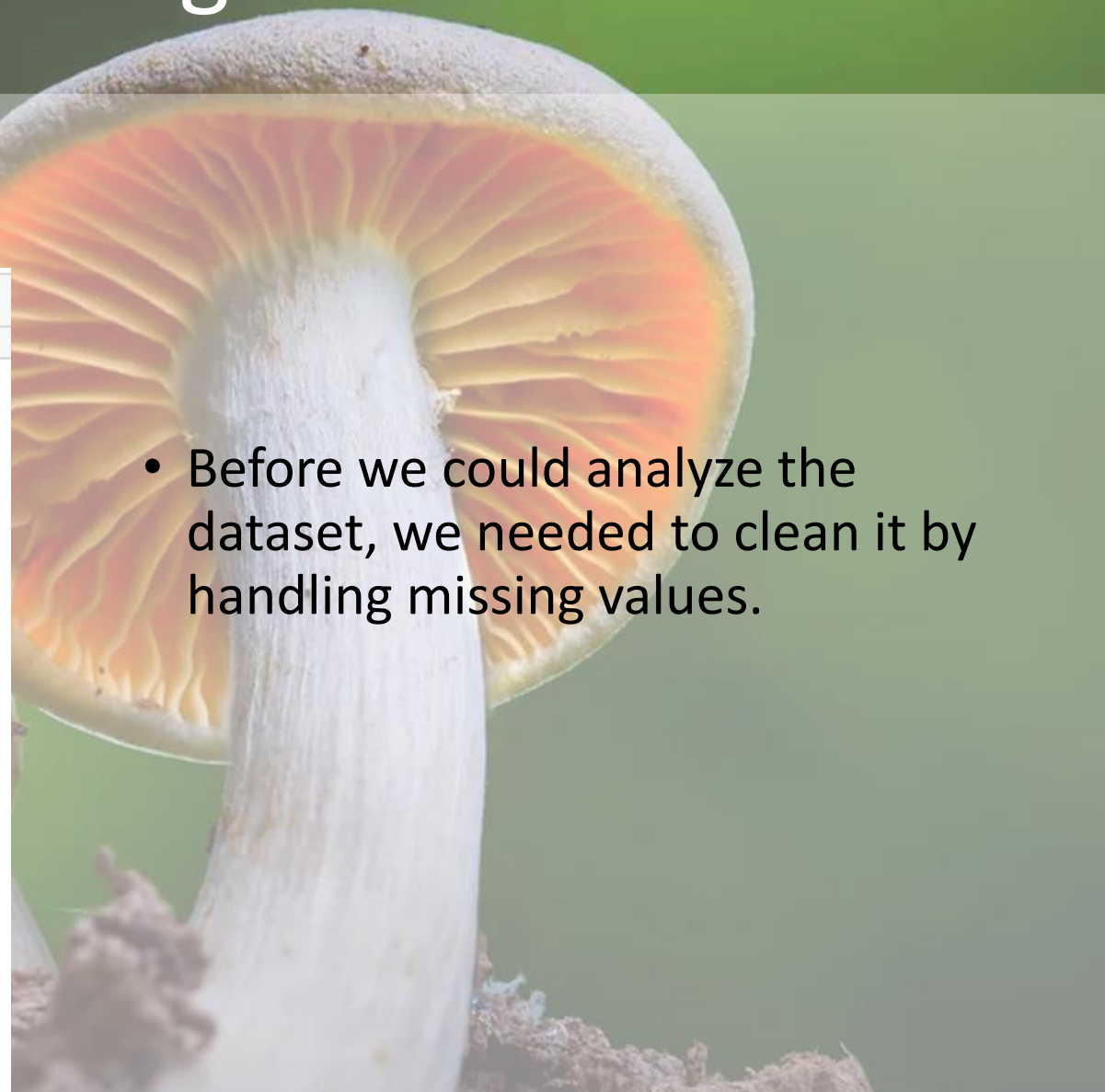
executed in 108ms, finished 14:58:56 2023-04-18

```
Out[8]: 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 [9]:  1 data.isna().sum()
```

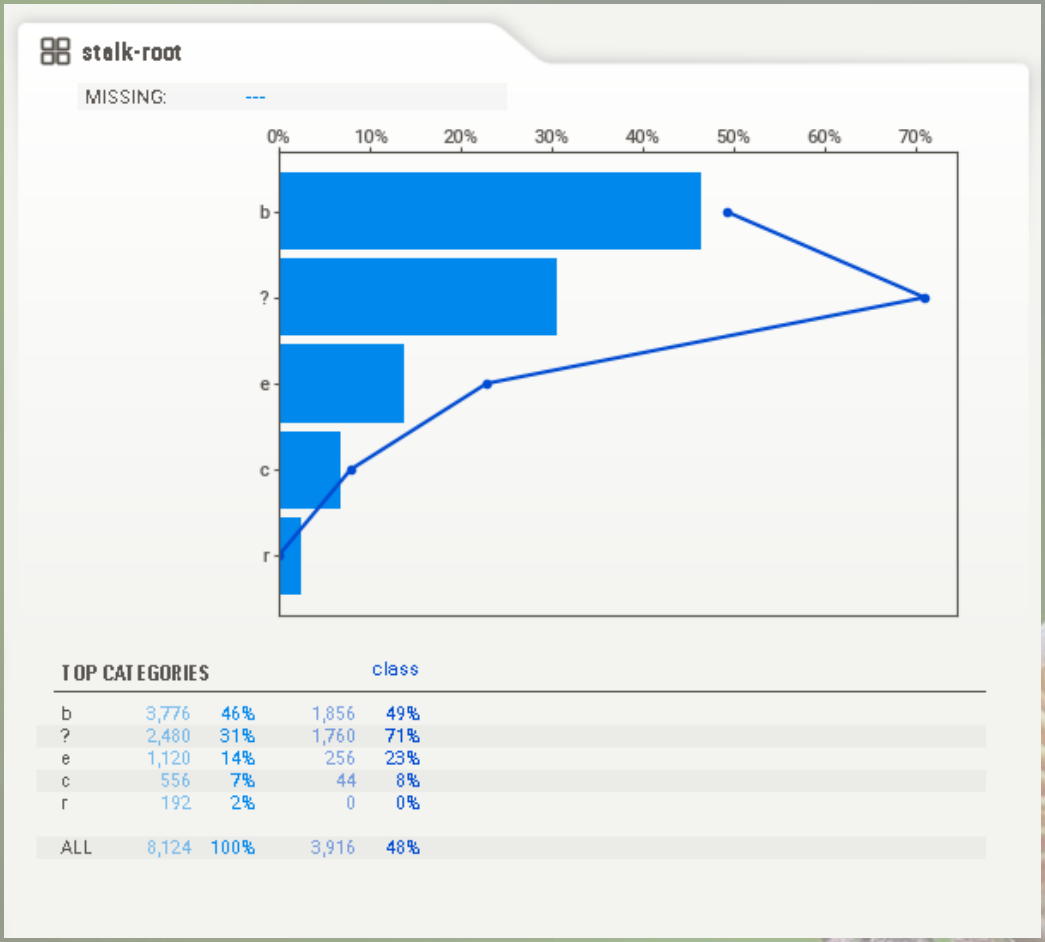
executed in 108ms, finished 14:58:56 2023-04-18

```
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
```



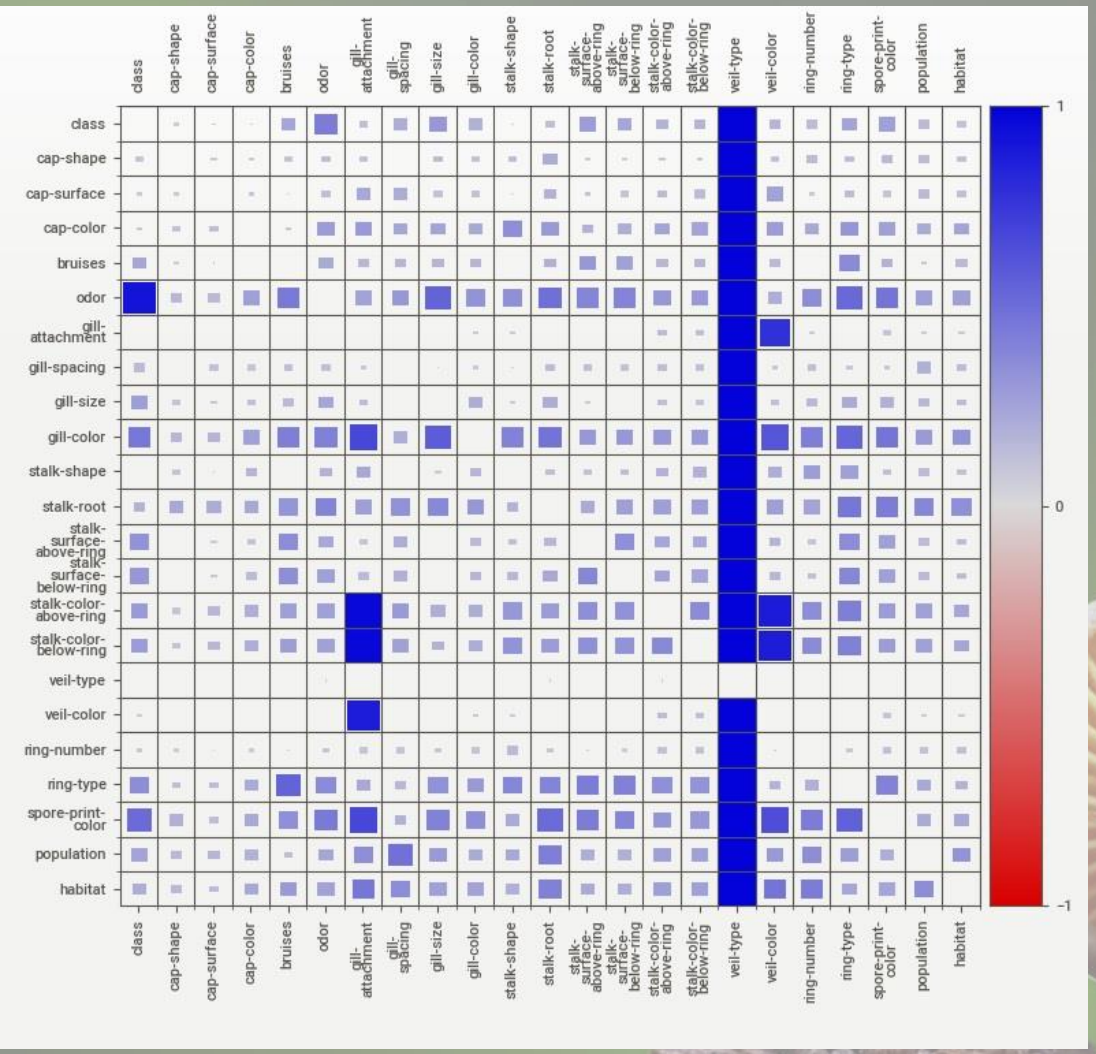
- Before we could analyze the dataset, we needed to clean it by handling missing values.

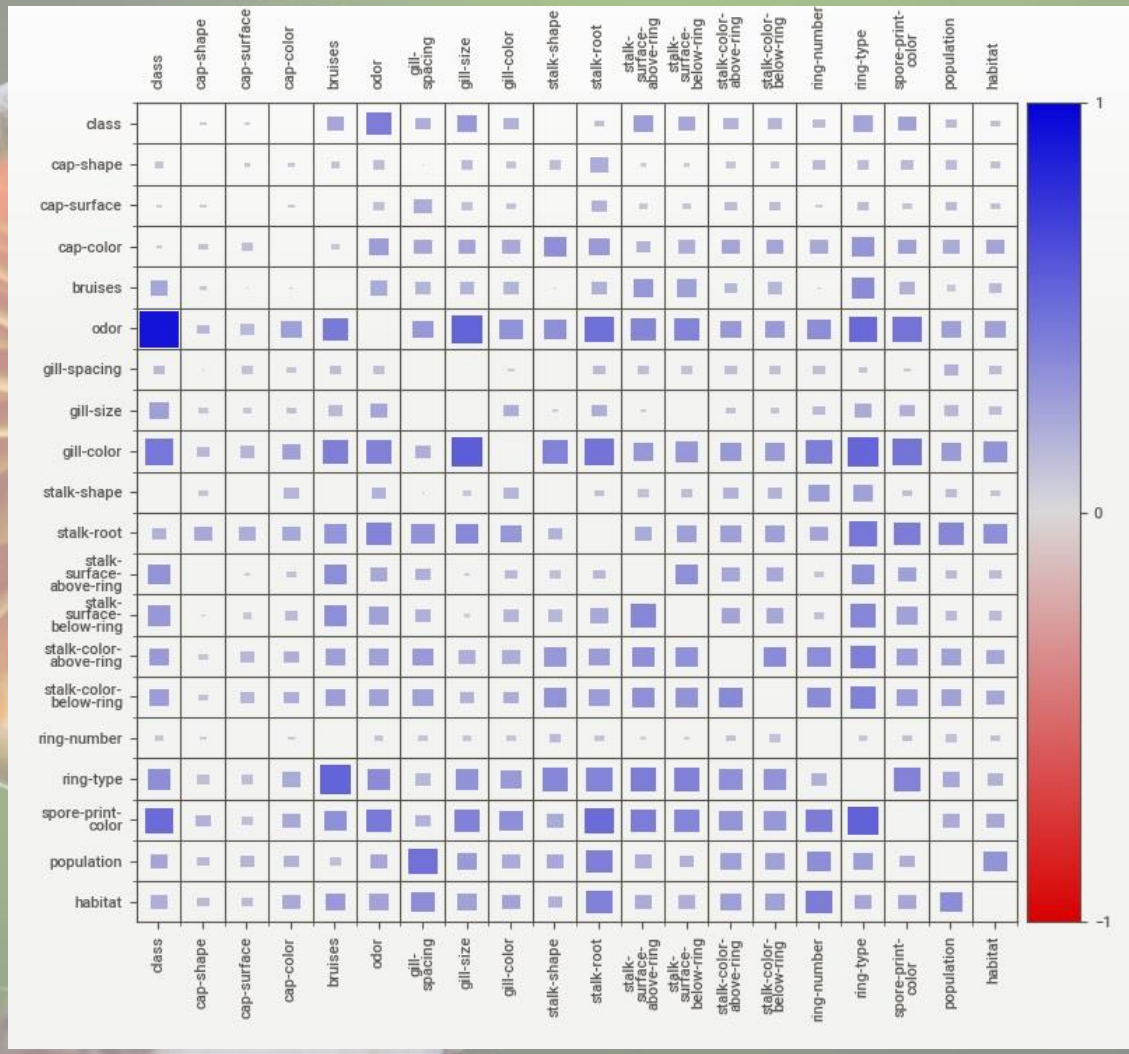
Missing values



Stock root variable has 2480 unknown values..
Applying imputation for this column would not be correct as the unknown values account for 31% of the data.
We will treat this as a different category called unknown.

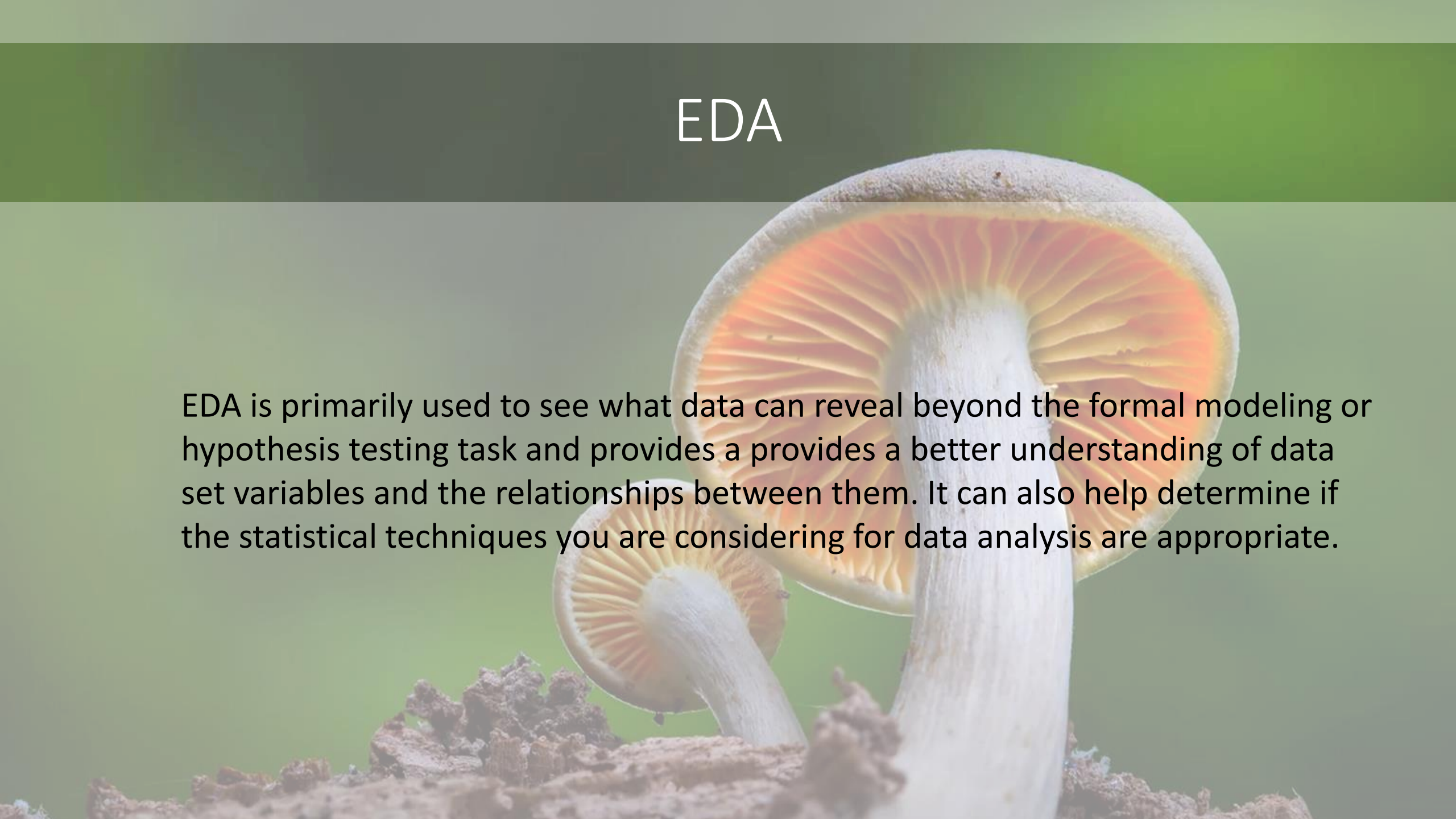
Correlation between variables





As veil-type, veil-color, gill-attachments have a very low correlation, we drop these from the data

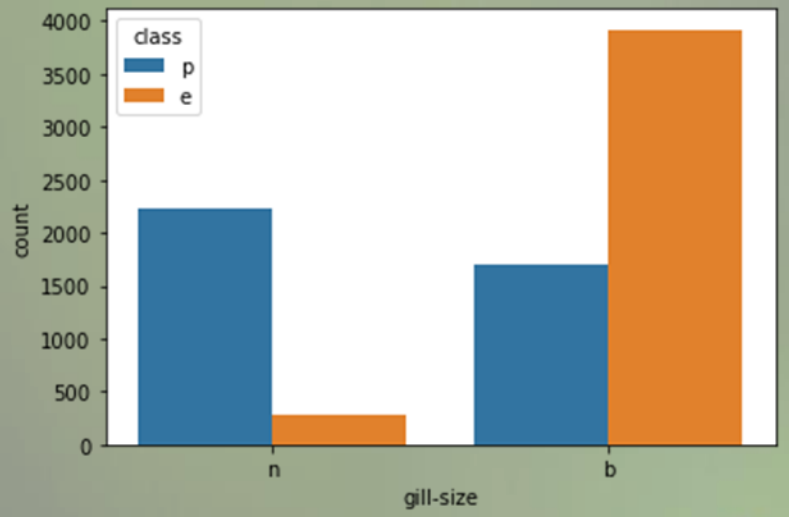
EDA

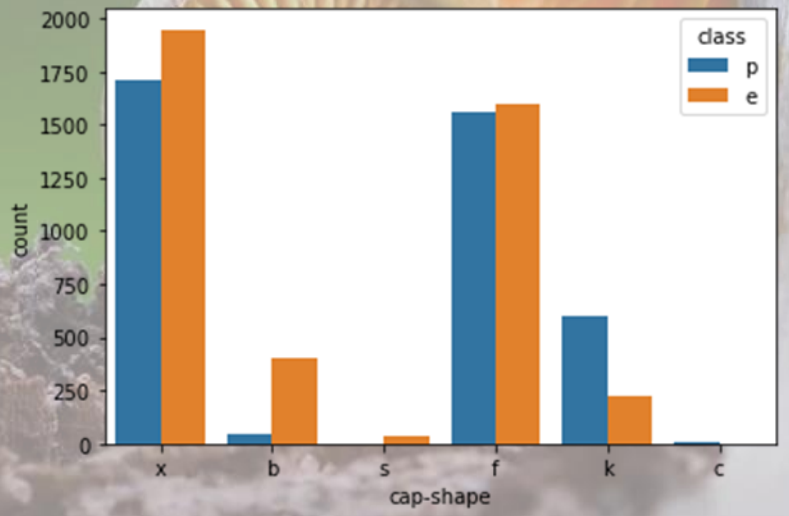


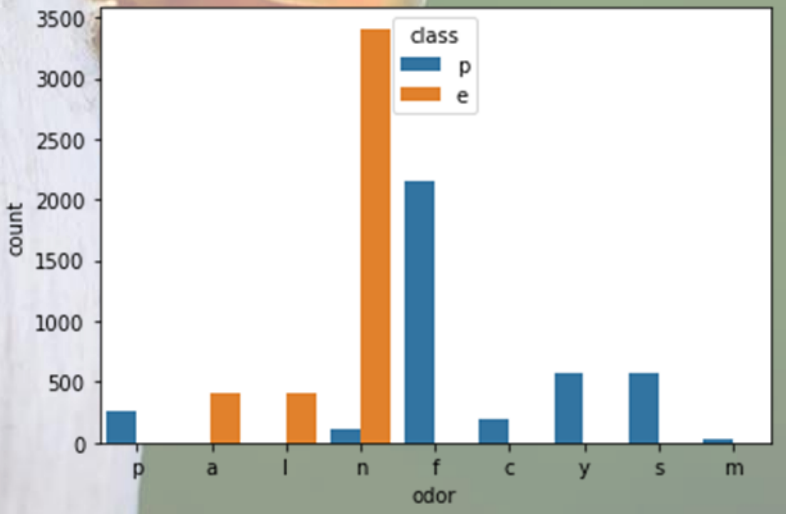
EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate.

EDA and Insights

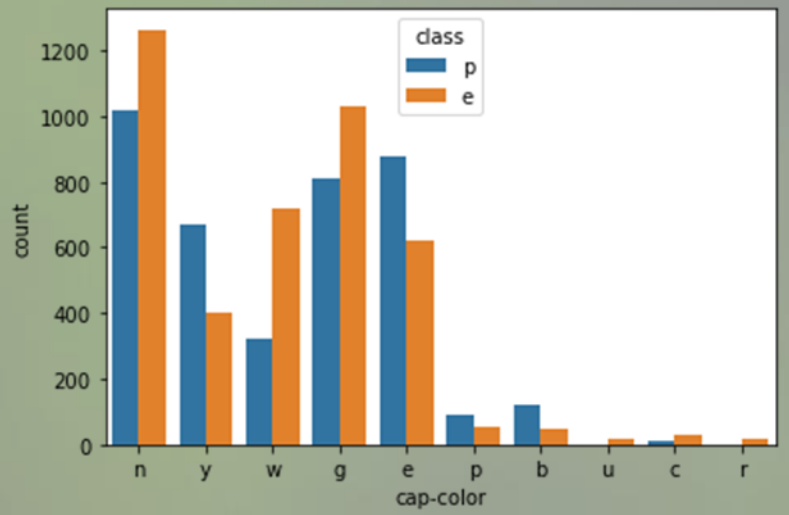
- Edible mushrooms mostly have a broad gill size and do not have conical caps.
- Edible mushrooms do not have any kind of bad like foul, fishy, pungent odor.

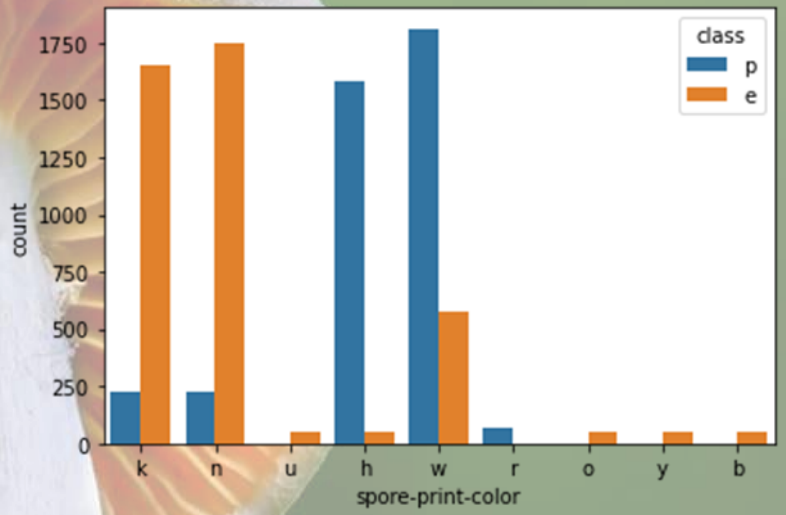


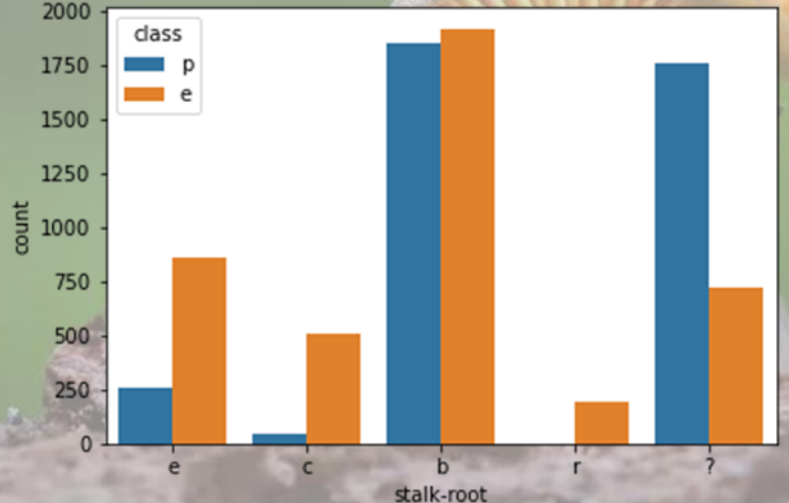


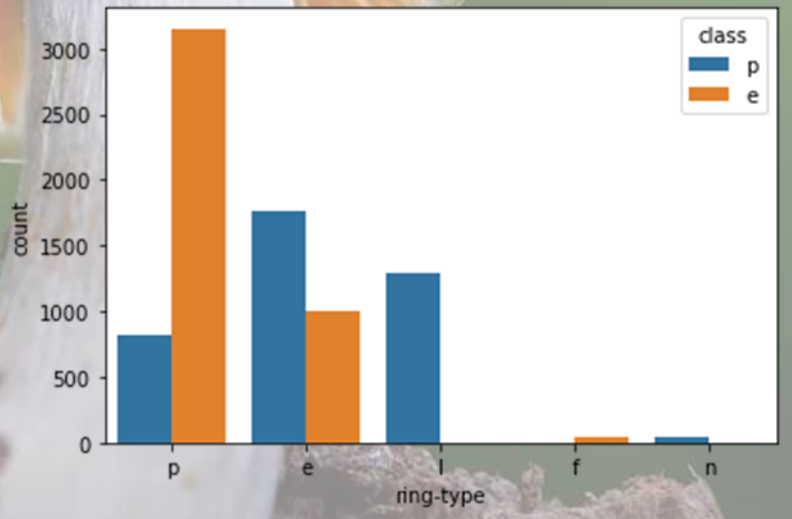


- Every red and orange colored cap of a mushroom is edible
- The stalk roots of edible mushrooms don't look like rhizomorphs.
- All flaring ring types are edible and large and none ring types are definitely poisonous.
- Mushrooms with the spore print buff, yellow, orange, and purple are always edible in nature. While green is for poisonous.

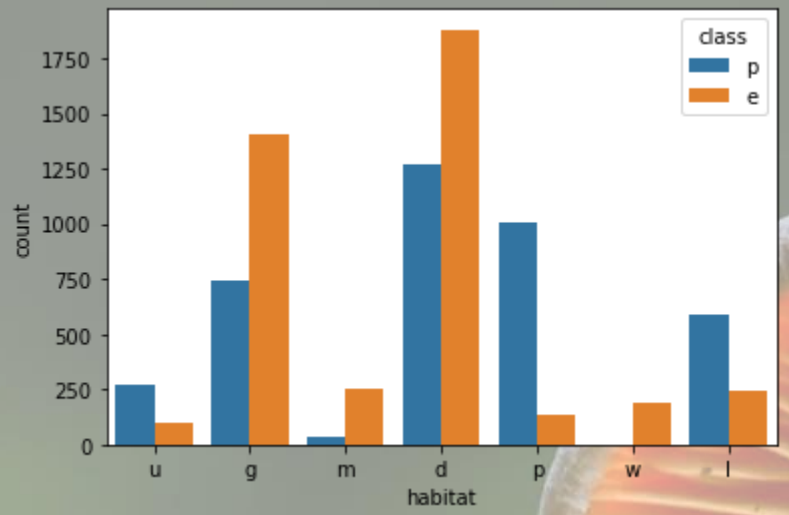




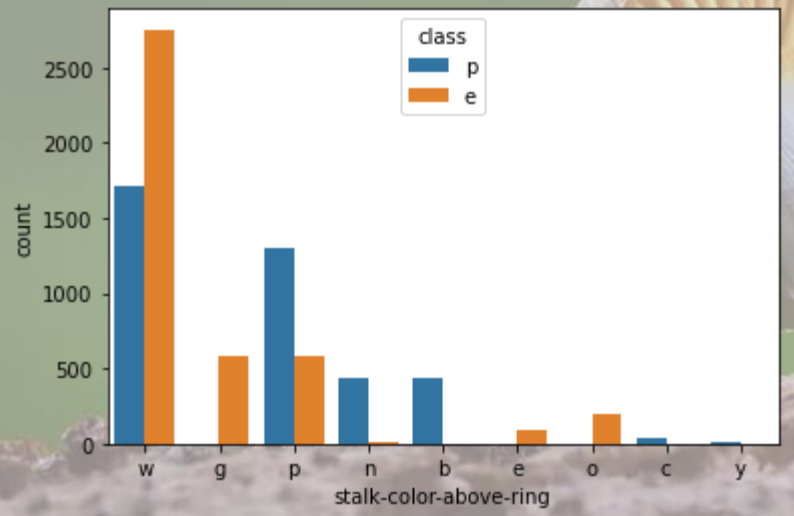


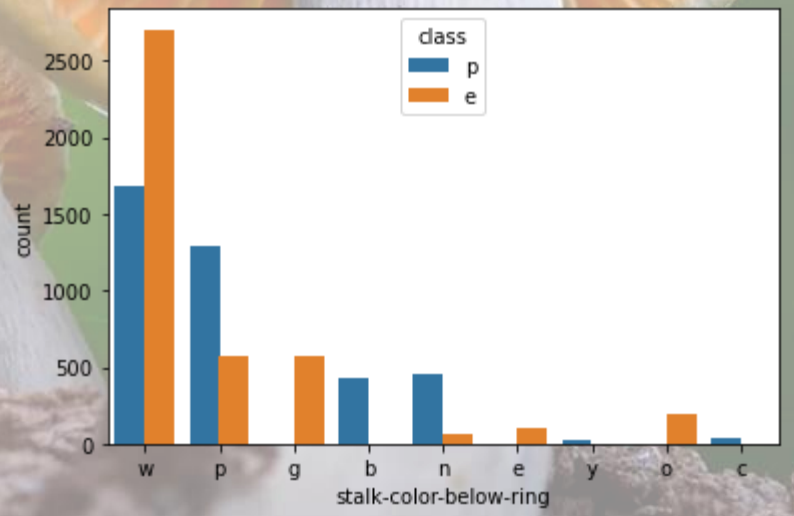


- Edible mushrooms can grow in all population types where as poisonous cannot grow in waste

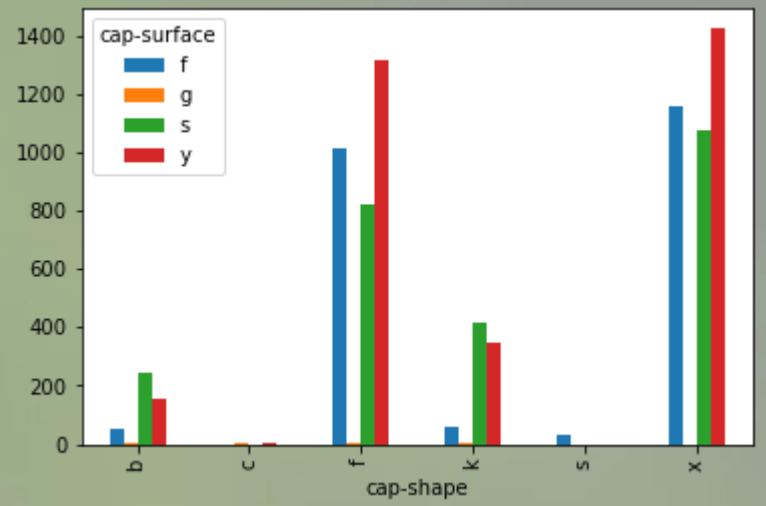


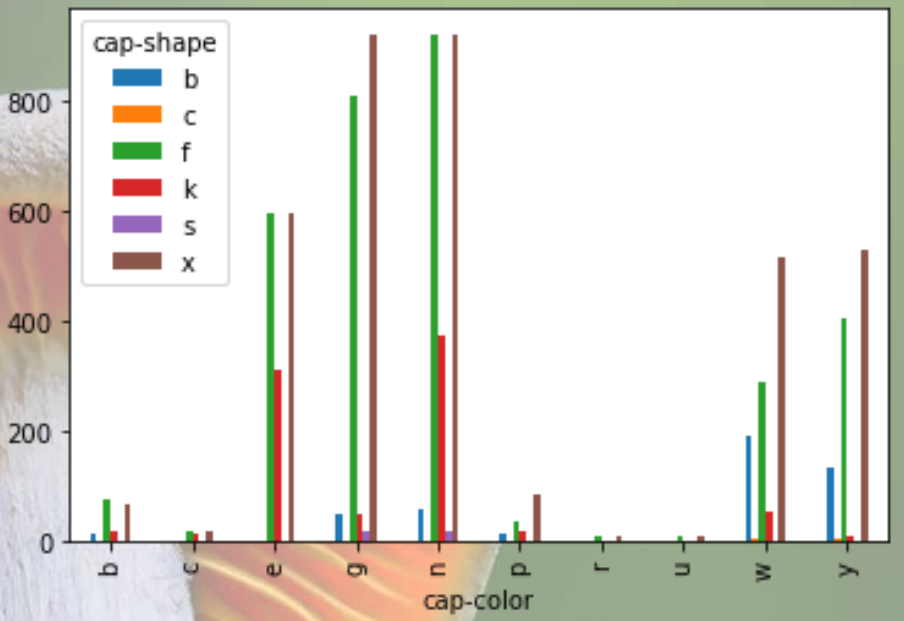
- If the stalk color above or below the ring is red, orange or grey it is definitely an edible mushroom. With stalk color cinnamon, yellow, buff it has to a poisonous mushroom.



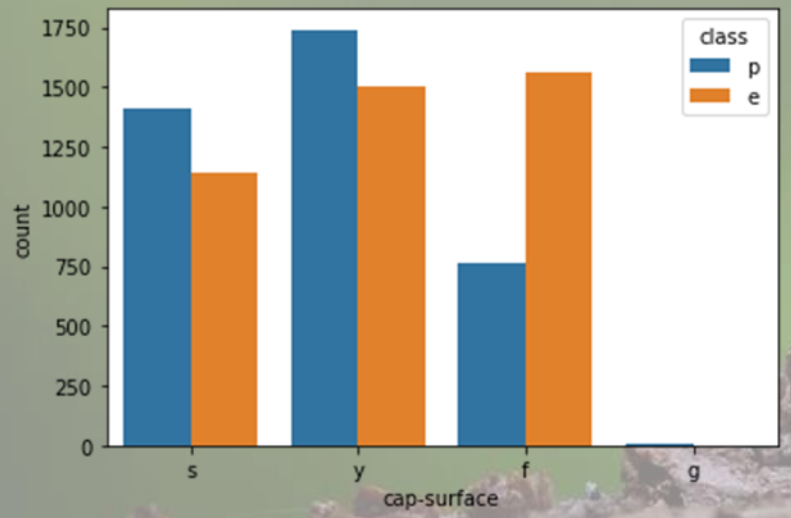


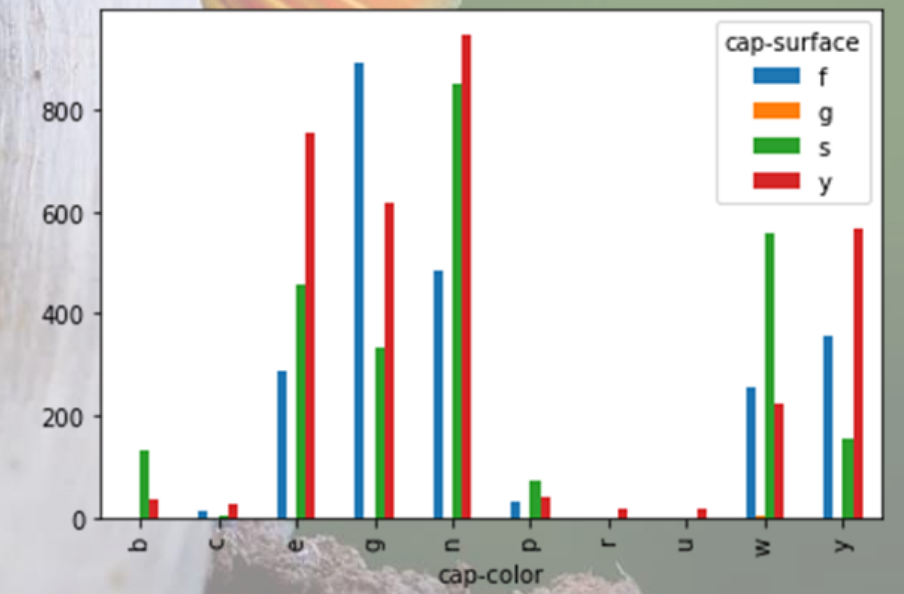
- If the cap shape is sunken then its surface is always fibrous.
- The mushrooms with conical-shaped caps with surface of cap as grooves are poisonous in nature.



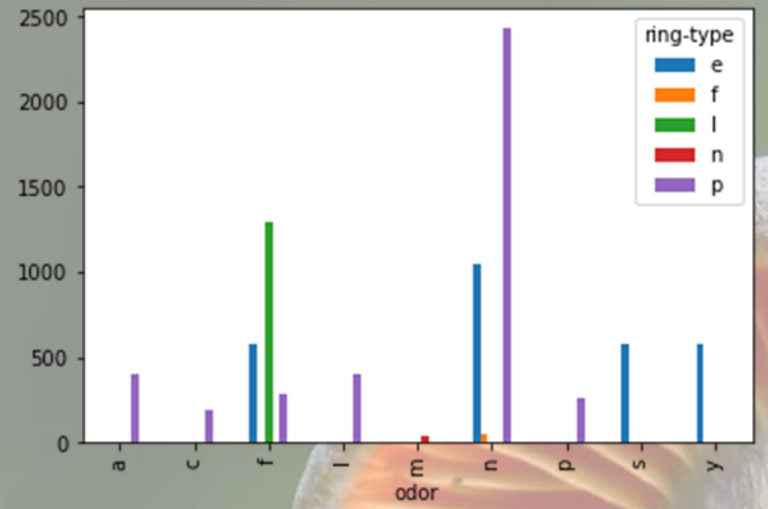


Conical cap-shaped mushrooms are only white and yellow, and Mushrooms whose cap surface is groove are only white in color.

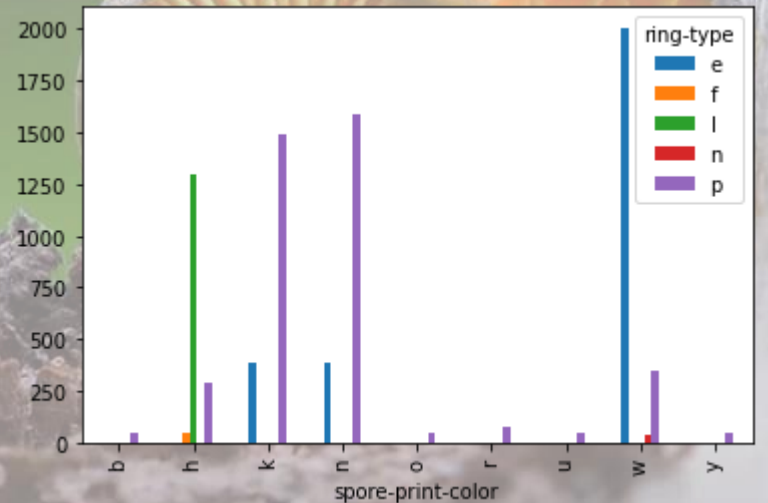




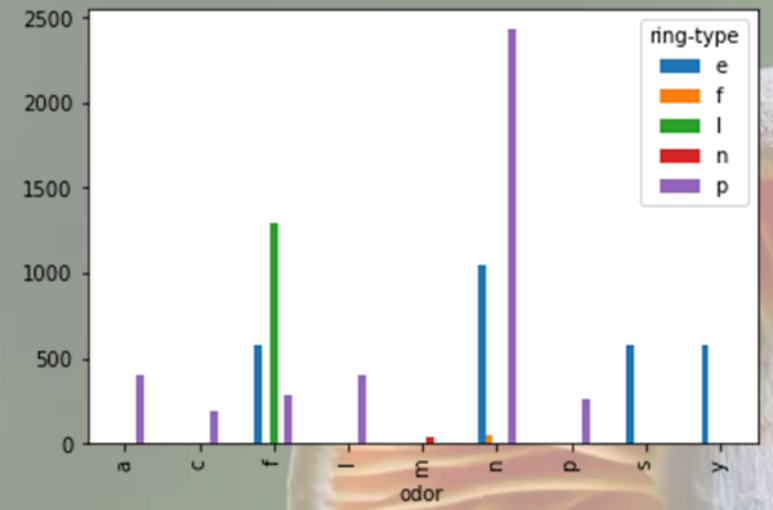
- The large ring type has only a foul odor and the Flaring ring type does not have an odor.



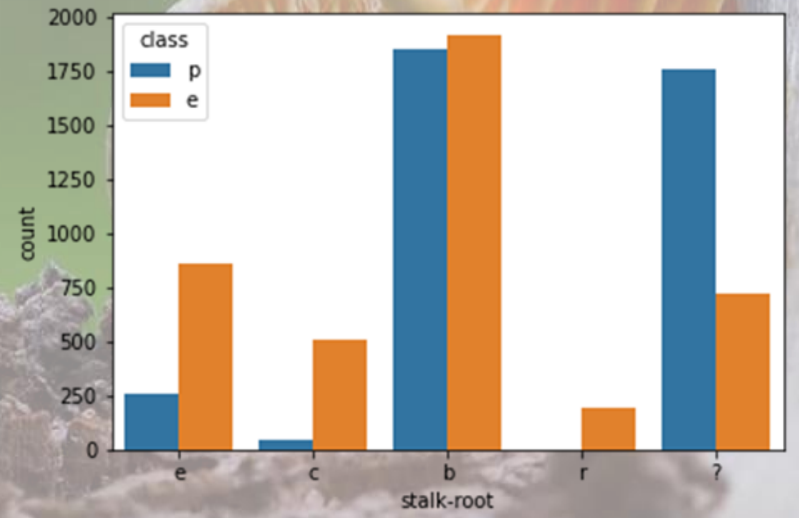
- Mushrooms which having large rings have a spore print color of chocolate. When there are no rings then the spore print color is white



- If the mushroom has a spicy and fishy odor then it will have an evanescent ring type. Whereas, almond, creosote, anise, and pungent odor have a pendant ring type.

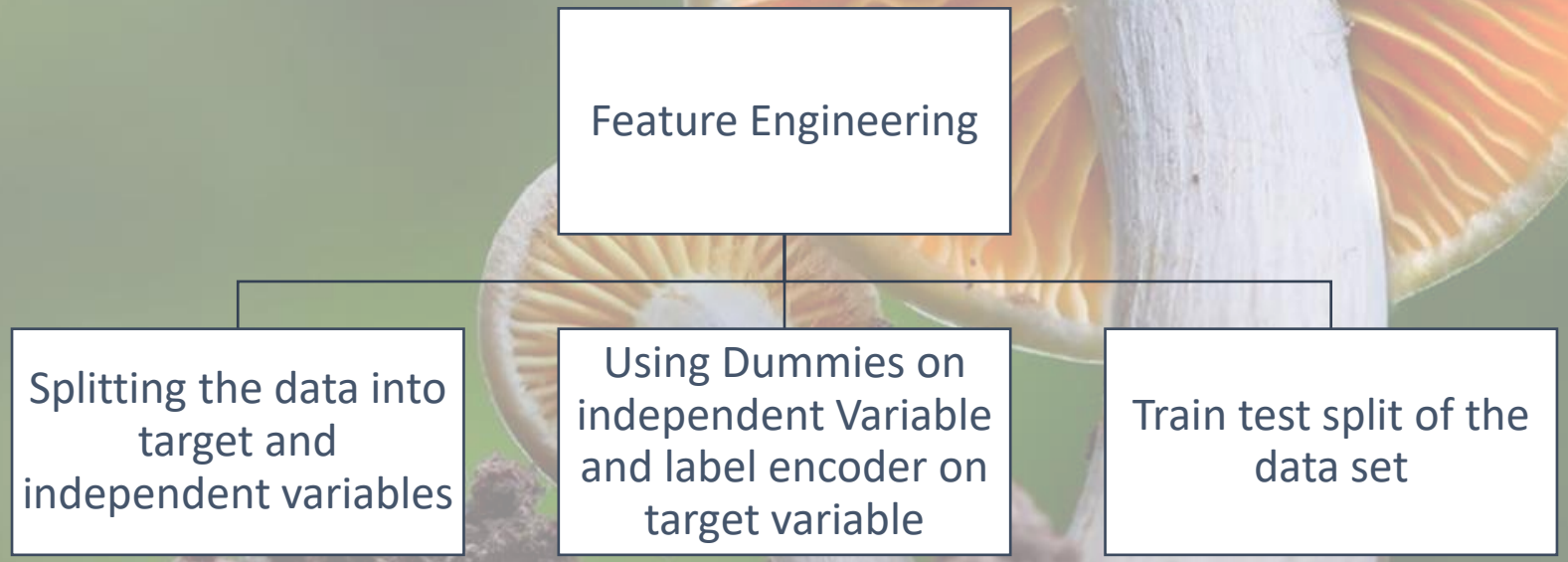


- Poisonous mushrooms aren't rooted.



Feature Engineering

- Feature engineering is the pre-processing step of machine learning, which is used to transform raw data into features that can be used for creating a predictive model using Machine learning or statistical Modelling. Feature engineering in machine learning aims to improve the performance of models.



Dummies

- The get_dummies function is used to convert categorical variables into dummy or indicator variables. A dummy or indicator variable can have a value of 0 or 1.

```
In [32]:
1 X = pd.get_dummies(data=x)
executed in 65ms, finished 22:39:31 2023-04-18

In [33]:
1 X
executed in 33ms, finished 22:39:33 2023-04-18
```

Out[33]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	...	population_s	population_v	population_y	habit
0	0	0	0	0	0	1	0	0	1	0	...	1	0	0	
1	0	0	0	0	0	1	0	0	1	0	...	0	0	0	
2	1	0	0	0	0	0	0	0	1	0	...	0	0	0	
3	0	0	0	0	0	1	0	0	0	1	...	1	0	0	
4	0	0	0	0	0	1	0	0	1	0	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8119	0	0	0	1	0	0	0	0	1	0	...	0	0	0	
8120	0	0	0	0	0	1	0	0	1	0	...	0	1	0	
8121	0	0	1	0	0	0	0	0	1	0	...	0	0	0	
8122	0	0	0	1	0	0	0	0	0	1	...	0	1	0	
8123	0	0	0	0	0	1	0	0	1	0	...	0	0	0	

8124 rows × 110 columns

Label Encoder

- Label Encoding is a technique that is used to convert categorical columns into numerical ones so that they can be fitted by machine learning models which only take numerical data.
- Most Machine Learning algorithms require numerical features. However, the dataset is composed of categorical features. We now must proceed to convert these to numerical data types.

```
In [34]:  1 le= LabelEncoder()
          executed in 14ms, finished 22:39:56 2023-04-18

In [35]:  1 Y=le.fit_transform(y)
          executed in 6ms, finished 22:40:09 2023-04-18

In [36]:  1 Y
          executed in 16ms, finished 22:40:11 2023-04-18

Out[36]: array([1, 0, 0, ..., 0, 1, 0])
```

- We are using label encoder for target variable (class)

Train test Split

Train test split A train test split is when you split your data into a training set and a testing set. The training set is used for training the model, and the testing set is used to test your model. We use 70% for training and 30% for testing. This ensures that both sets are representative of the entire dataset.

Train dataset: used to fit the machine learning model

Test Dataset: is used to evaluate the fit machine learning model .

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

```
In [38]:
1 from sklearn.model_selection import train_test_split
executed in 8ms, finished 22:41:32 2023-04-18

In [39]:
1 x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.3)
executed in 34ms, finished 22:42:13 2023-04-18

In [40]:
1 x_train
executed in 39ms, finished 22:42:22 2023-04-18
```

Out[40]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	...	population_s	population_v	population_y	habit
5837	0	0	0	0	0	1	0	0	0	1	...	0	1	0	
1535	0	0	0	0	0	1	0	0	1	0	...	1	0	0	
2991	0	0	0	0	0	1	1	0	0	0	...	0	1	0	
297	0	0	0	0	0	1	1	0	0	0	...	1	0	0	
2433	0	0	0	0	0	1	1	0	0	0	...	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2656	0	0	0	0	0	1	0	0	0	1	...	0	0	1	
5869	0	0	0	0	0	1	0	0	0	1	...	0	1	0	
2096	0	0	1	0	0	0	1	0	0	0	...	0	1	0	
5089	0	0	1	0	0	0	0	0	0	1	...	0	0	1	
3189	0	0	1	0	0	0	1	0	0	0	...	0	0	1	

5686 rows × 110 columns

```
In [41]:
1 x_test
executed in 29ms, finished 22:42:36 2023-04-18
```

Out[41]:

	cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_f	cap-surface_g	cap-surface_s	cap-surface_y	...	population_s	population_v	population_y	habit
5133	0	0	0	0	0	1	0	0	0	1	...	0	1	0	
3444	0	0	0	0	0	1	0	0	1	0	...	1	0	0	
5997	0	0	0	0	0	1	0	0	0	1	...	0	0	1	
7626	0	0	0	0	0	1	0	0	1	0	...	1	0	0	
6614	0	0	1	0	0	0	0	0	1	0		0	1	0	