3/14/2021

Identification of Poisonous Mushrooms based on their physical features- AN ML BASED APPROACH

Poojitha Bhat

Introduction:

We grew up playing Mario and seeing him power up from mushrooms in mushroom kingdom. Didn't we? Mario loved to munch on different super mushrooms in his kingdom and grow invincible. We also remember the Poison shroom that would trap Mario and cause him to lose his abilities. Much like Mario's Mushroom Kingdom, the Mushrooms found on planet Earth are diverse. The colors ranging from white to blue, mushrooms wearing skirts to the ones wearing veils, plain to radio- active; the diversity is rich. The edible ones are rich sources of Vitamin B and can add on to a healthy diet. These beautiful fungi can also be poisonous being disguised to resemble their healthy counterparts. These poisonous mushrooms are often mis-identified as edible and ingested by humans. The consequences may range from stomach upset to death. Around 7500 mushroom poisoning cases are reported in the USA every year. [1]

Bringing Machine learning to the rescue, the aim of this project is to identify the poisonous varieties of mushrooms based on their physical features such as color, odor, gills, stalk size, stalk type and others. With the model we build, we hope to help Mario be undeterred in his mission to rescue Princess Peach!

Tools used:

Python 3.7.1
Tableau Public 2020.4

Libraries used:

Pandas
Numpy
Matplotlib
Sklearn

Data exploration and cleansing:

The dataset was obtained from UCI Machine Learning repository [2] in CSV format.

```
mushroom = pd.read_csv('Mushroom.csv')
mushroom.shape
```

```
(8124, 23)
```

The CSV file was imported using pandas. The data had 8124 data points and 23 mushroom features.

```
mushroom.sample(10)
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color
7187	p	k	y	n	f	s	f	c	n	b
2544	e	x	y	g	t	n	f	c	b	w
7006	p	k	s	e	f	f	f	c	n	b
6843	p	k	y	e	f	s	f	c	n	b
3214	e	f	f	e	t	n	f	c	b	n
1145	e	x	f	w	f	n	f	w	b	n
2034	e	x	y	g	t	n	f	c	b	u
3833	p	x	s	w	f	c	f	c	n	n
5947	p	b	s	b	t	n	f	c	b	r
888	e	x	y	y	t	a	f	c	b	w

10 rows × 23 columns

On sampling 10 data points it was seen that the features were encoding using letters. This makes the dataset difficult to comprehend.

On a high level we see features such as cap shape/size/color, gill attachment/spacing etc. The first column is the mushroom class. We see two classes mostly standing for Edible and Poisonous.

```
mushroom.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)
```

All features seem to have non-null values with unique identifiers.

However the presence of alphabet coding makes it difficult to explore the data further. These have to be replaced with their actual names.

```
#creating a dictionary with readable names

dictionary = {'class' : {'e': 'edible', 'p': 'poisonous'} ,
           'cap-shape': {'b':'bell', 'c':'conical', 'x': 'convex' , 'f':'flat', 'k':'knobbed' , 's':'sunken'},
          'cap-surface':{'f':'fibrous','g':'grooves','y':'scaly', 's' :'smooth'},
          'cap-color': {'n':'brown','b':'buff','c':'cinnamon', 'g': 'gray', 'r':'green','p':'pink','u': 'purple', 'e': 'red',
          'bruises': {'t':'bruises','f':'no'},
          'odor': {'a':'almond','l' :'anise','c':'creosote','y':'fishy','f': 'foul','m':'musty','n' :'none','p':'pungent','s
          'gill-attachment': {'a':'attached','d':'descending','f':'free','n':'notched'},
          'gill-spacing': {'c':'close','w':'crowded','d':'distant'},
          'gill-size': {'b':'broad','n':'narrow'},
          'gill-color': {'k':'black','n':'brown','b':'buff','h': 'chocolate','g':'gray','r': 'green', 'o' :'orange', 'p': 'pi
          'stalk-shape': {'e':'enlarging','t':'tapering'},
           'stalk-root': {'b':'bulbous','c':'club','u':'cup','e':'equal','z':'rhizomorphs','r':'rooted','?':'missing'},
           'stalk-surface-above-ring' : {'f':'fibrous','y':'scaly','k':'silky','s':'smooth'},
           'stalk-surface-below-ring': {'f':'fibrous','y':'scaly','k':'silky','s':'smooth'},
           'stalk-color-above-ring': {'n':'brown','b':'buff','c':'cinnamon', 'g' :'gray','o':'orange','p':'pink','e':'red','w
           'stalk-color-below-ring': {'n':'brown','b':'buff','c':'cinnamon', 'g' :'gray','o':'orange','p':'pink','e':'red','w
           'veil-type': {'p':'partial','u':'universal'},
           'veil-color': {'n': 'brown','o' :'orange','w': 'white', 'y':'yellow'},
           'ring-number': {'n':'none','o':'one','t' : 'two'},
           'ring-type': {'c':'cobwebby','e':'evanescent','f':'flaring','l': 'large','n':'none','p':'pendant','s':'sheathing',
           'spore-print-color': {'k':'black','n':'brown','b':'buff','h': 'chocolate','g':'gray','r': 'green', 'o' :'orange',
           'population': {'a':'abundant','c':'clustered','n':'numerous','s':'scattered','v':'several','y':'solitary'},
           'habitat': {'g':'grasses','l':'leaves','m':'meadows','p':'paths','u':'urban','w':'waste','d':'woods'}


}
mushroom_renamed = mushroom.replace(dictionary)
```

A dictionary created with all the columns and their letter encoding. The values in the data have been replaced with these values. Let's sample a few data points to see if it has worked.

```
mushroom_renamed.sample(10)
```

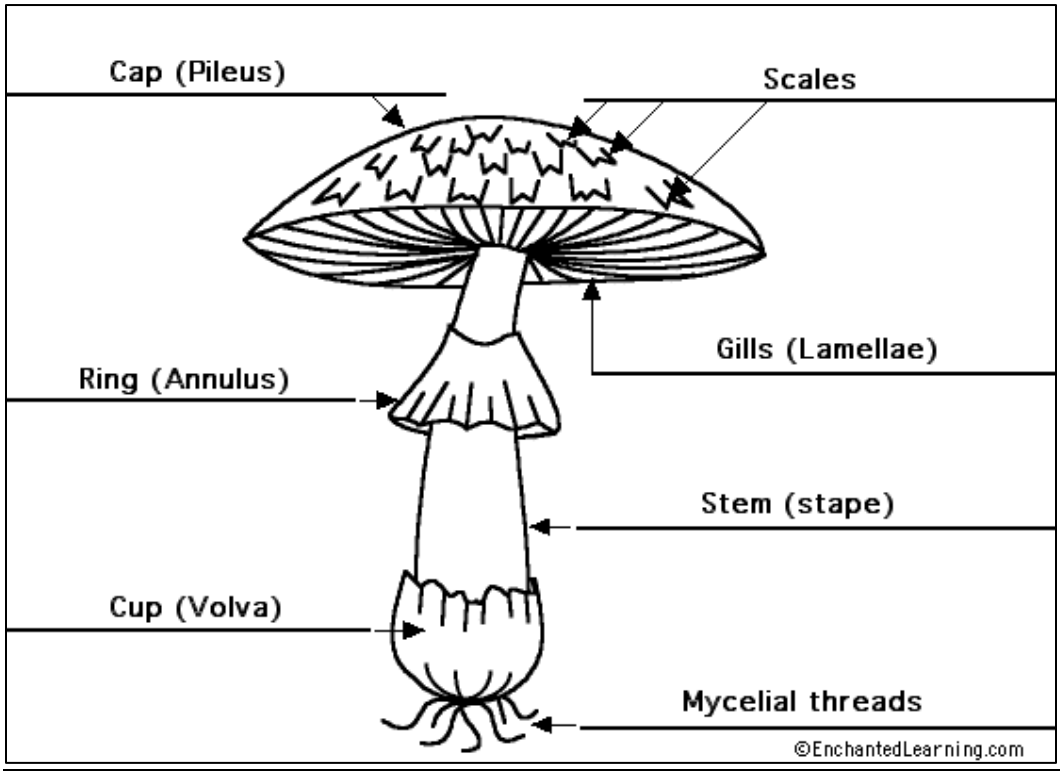
	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color
6266	poisonous	flat	scaly	brown	no	spicy	free	close	narrow	buff
537	edible	sunken	fibrous	brown	no	none	free	close	narrow	pink
3136	edible	flat	fibrous	red	bruises	none	free	close	broad	pink
6243	poisonous	flat	scaly	red	no	fishy	free	close	narrow	buff
6539	poisonous	flat	smooth	red	no	fishy	free	close	narrow	buff
4442	poisonous	convex	scaly	yellow	no	foul	free	close	broad	pink
2267	edible	convex	scaly	brown	bruises	none	free	close	broad	pink
3900	poisonous	convex	fibrous	gray	no	foul	free	close	broad	pink
4274	poisonous	flat	scaly	gray	no	foul	free	close	broad	pink
5070	edible	convex	scaly	white	no	none	free	close	narrow	purple

10 rows × 23 columns

The dictionary has worked at replacing the coded values. We now have a dataset whose features can easily be explored.

These features will be explored one by one in the next step.

Analysis of features:



A mushroom typically consists of a cap with/without scales, a stem with a ring and roots in the form of mycelia. The dataset we contain needs to be explored for these features and if they correlate to scientific literature.

Class:

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

edible       4208
poisonous    3916
```

There are only two classes of mushrooms seen: Edible and Poisonous. This will be our outcome/dependent feature and will be crucial in training our model.

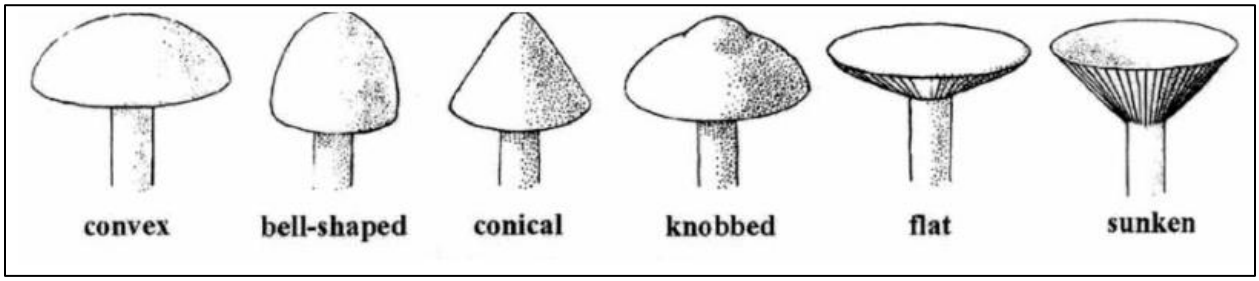
We see a roughly equal number of both the classes. This means that the dataset is balanced and there will be no issues training the model.

Cap shape:

```
mushroom_renamed['cap-shape'].groupby(mushroom_renamed['cap-shape']).count()
```

```
cap-shape
bell        452
conical       4
convex     3656
flat       3152
knobbed     828
sunken       32
```

There are 6 different shapes seen. Below are the commonly found cap shapes of mushrooms as per scientific literature.

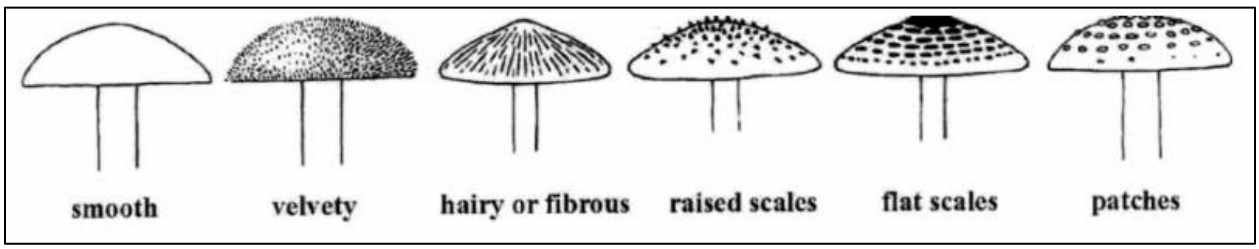


Cap surface:

```
mushroom_renamed['cap-surface'].value_counts()
```

```
scaly      3244
smooth     2556
fibrous    2320
grooves       4
Name: cap-surface, dtype: int64
```

There are four different cap surfaces seen. Scaly,smooth and fibrous have roughly equal representation. There are 4 mushrooms with grooved surface. Grooved surface here represents the patchy mushrooms as per the literature.



Cap Color:

```
mushroom_renamed['cap-color'].groupby(mushroom_renamed['cap-color']).count()
```

```
cap-color
brown      2284
buff        168
cinnamon     44
gray       1840
green        16
pink        144
purple       16
red        1500
white      1040
yellow     1072
```

Mushrooms have a colorful life! We have 10 different colors of mushrooms on our data. There are more colors seen in the wild.

Surface Bruises:

```
mushroom_renamed['bruises'].groupby(mushroom_renamed['bruises']).count()
```

```
bruises
bruises    3376
no         4748
```

The mushrooms can either have bruises or not have them.

Mushroom Odor:

```
mushroom_renamed['odor'].groupby(mushroom_renamed['odor']).count()
```

```
odor
almond       400
anise        400
creosote     192
fishy        576
foul        2160
musty         36
none        3528
pungent      256
spicy        576
```

Most of the mushrooms have no odor. However half of them have different types of odors.

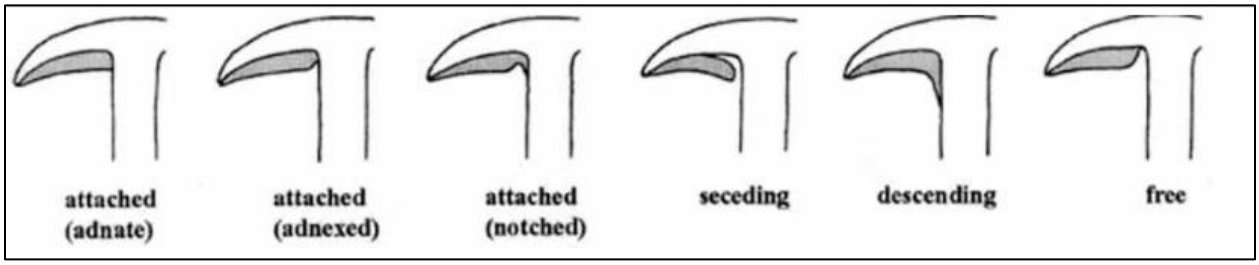
Gill Attachment:

```
mushroom_renamed['gill-attachment'].groupby(mushroom_renamed['gill-attachment']).count()
```

```
gill-attachment
attached     210
free        7914
```

The gills can either be attached or free.

Below are the pictures of different types of Gill attachment



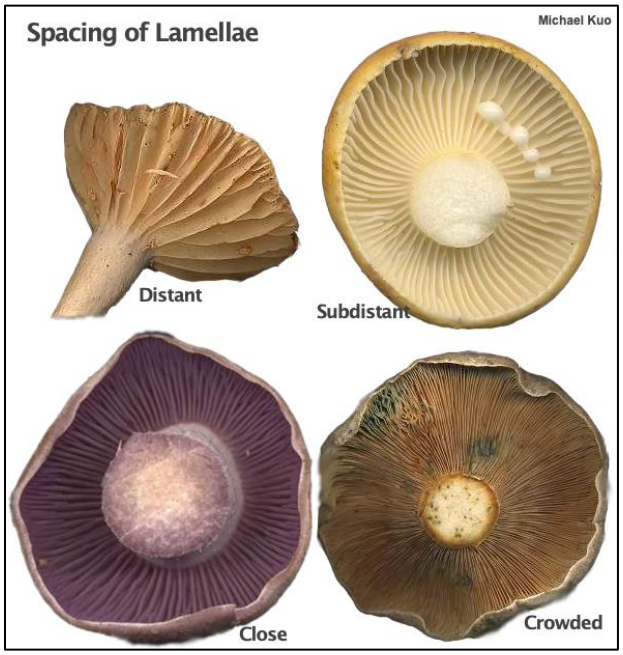
In our data all the different types of attachment seen above are categorized under ‘attached’. Most of the mushrooms have free gills.

Gill Spacing:

```
mushroom_renamed['gill-spacing'].groupby(mushroom_renamed['gill-spacing']).count()
```

```
gill-spacing
close      6812
crowded    1312
```

Our mushrooms have either closely spaced gills or crowded gills.



We don’t have the mushrooms with Distant and Sub distant gill spacing in our dataset.

Gill size:

```
mushroom_renamed['gill-size'].groupby(mushroom_renamed['gill-size']).count()
```

```
gill-size
broad     5612
narrow    2512
```

The gills can be broad or narrow

Gill color:

```
mushroom_renamed['gill-color'].groupby(mushroom_renamed['gill-color']).count()
```

```
gill-color
black         408
brown        1048
buff         1728
chocolate     732
gray          752
green          24
orange         64
pink         1492
purple        492
red            96
white        1202
yellow         86
```

There are 10 different types of gill colors seen. They seem to be similar to the mushroom colors seen previously.

Stalk Shape:

```
mushroom_renamed['stalk-shape'].groupby(mushroom_renamed['stalk-shape']).count()
```

```
stalk-shape
enlarging    3516
tapering     4608
```

The stalks can either be enlarging or they can be tapering.

Stalk Root:

```
mushroom_renamed['stalk-root'].groupby(mushroom_renamed['stalk-root']).count()
```

```
stalk-root
bulbous    3776
club        556
equal      1120
missing    2480
rooted      192
```

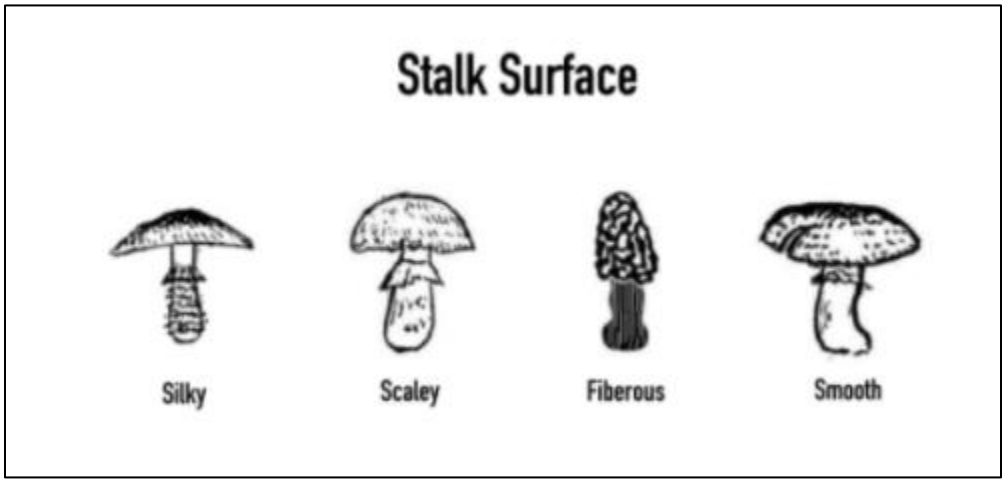
There are 2480 cases with missing stalks roots. Let’s consider these mushrooms to have no stalk roots.

Stalk Surface Above Ring:

```
mushroom_renamed['stalk-surface-above-ring'].groupby(mushroom_renamed['stalk-surface-above-ring']).count()
```

```
stalk-surface-above-ring
fibrous      552
scaly         24
silky       2372
smooth      5176
```

There are four types of stalk surfaces seen above the ring



Stalk Surface Below Ring:

```
mushroom_renamed['stalk-surface-below-ring'].groupby(mushroom_renamed['stalk-surface-below-ring']).count()
```

```
stalk-surface-below-ring
fibrous      600
scaly        284
silky       2304
smooth      4936
```

There are four types of stalk surfaces seen below ring also. But their distribution seems to be varied. Hence we don't have the same types of surfaces above and below ring.

Stalk Color Above Ring:

```
mushroom_renamed['stalk-color-above-ring'].groupby(mushroom_renamed['stalk-color-above-ring']).count()
```

```
stalk-color-above-ring
brown         448
buff          432
cinnamon       36
gray          576
orange        192
pink         1872
red            96
white        4464
yellow          8
```

There are 9 stalk colors seen above the ring.

Stalk Color Below Ring:

```
mushroom_renamed['stalk-color-below-ring'].groupby(mushroom_renamed['stalk-color-below-ring']).count()
```

```
stalk-color-below-ring
brown         512
buff          432
cinnamon       36
gray          576
orange        192
pink         1872
red            96
white        4384
yellow         24
```

There are 9 stalk colors seen below the ring as well. However the distribution is varied. Hence we don't have the same values as Stalk surface above ring.

Veil Type:

```
mushroom_renamed['veil-type'].groupby(mushroom_renamed['veil-type']).count()
```

```
veil-type
partial    8124
```

There is only one type of veil seen in the mushrooms. All of them have a partial veil.

Veil Color:

```
mushroom_renamed['veil-color'].groupby(mushroom_renamed['veil-color']).count()
```

```
veil-color
brown        96
orange       96
white      7924
yellow        8
```

4 different types of veil colors seen. No missing values found.

Number of Rings:

```
mushroom_renamed['ring-number'].groupby(mushroom_renamed['ring-number']).count()
```

```
ring-number
none        36
one       7488
two        600
```

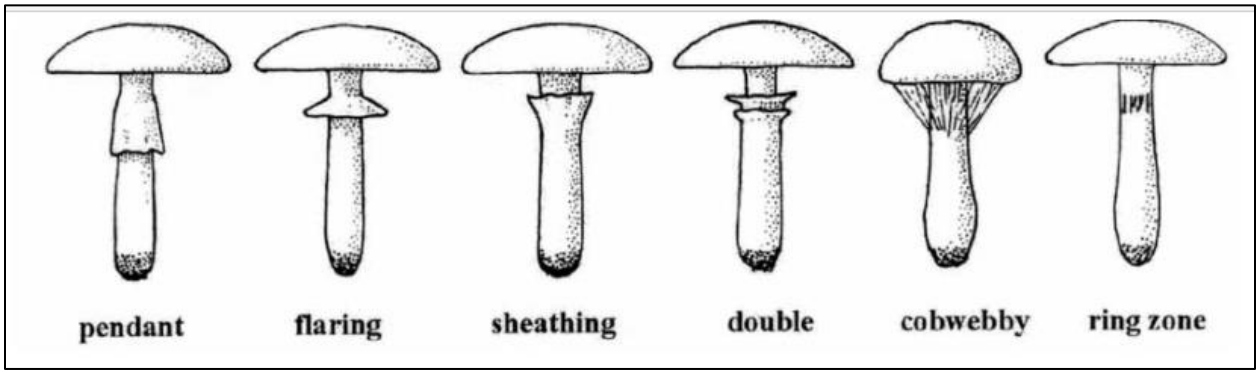
The mushrooms can have 1, 2 or no rings at all.

Ring Type:

```
mushroom_renamed['ring-type'].groupby(mushroom_renamed['ring-type']).count()
```

```
ring-type
evanescent    2776
flaring         48
large         1296
none            36
pendant       3968
```

The ring types can be evanescent, flaring, large or pendant type. This value is correctly missing for the 36 mushrooms that don't have rings at all.



Spore Print Color:

```
mushroom_renamed['spore-print-color'].groupby(mushroom_renamed['spore-print-color']).count()
```

```
spore-print-color
black        1872
brown        1968
buff           48
chocolate    1632
green          72
orange         48
purple         48
white        2388
yellow         48
```

Spores play a pivotal role in spreading the mushroom population and can of different colors as seen above.

Mushroom Population:

```
mushroom_renamed['population'].groupby(mushroom_renamed['population']).count()
```

```
population
abundant      384
clustered     340
numerous      400
scattered    1248
several      4040
solitary     1712
```

The mushrooms can be found all alone or in groups. Our data has mushrooms that are several in nature or are solitary or in clustered states.

Mushroom Habitat:

```
mushroom_renamed['habitat'].groupby(mushroom_renamed['habitat']).count()
```

```
habitat
grasses    2148
leaves      832
meadows     292
paths      1144
urban       368
waste       192
woods      3148
```

Mushrooms are mostly found in woods. But we are also seeing some of them grow with grasses in the meadows and urban areas too.

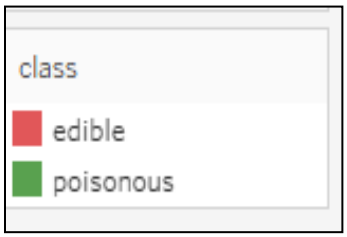
It's seen that the features are all correctly classified by their categories and also correlate to scientific literature. We can proceed with further analysis.

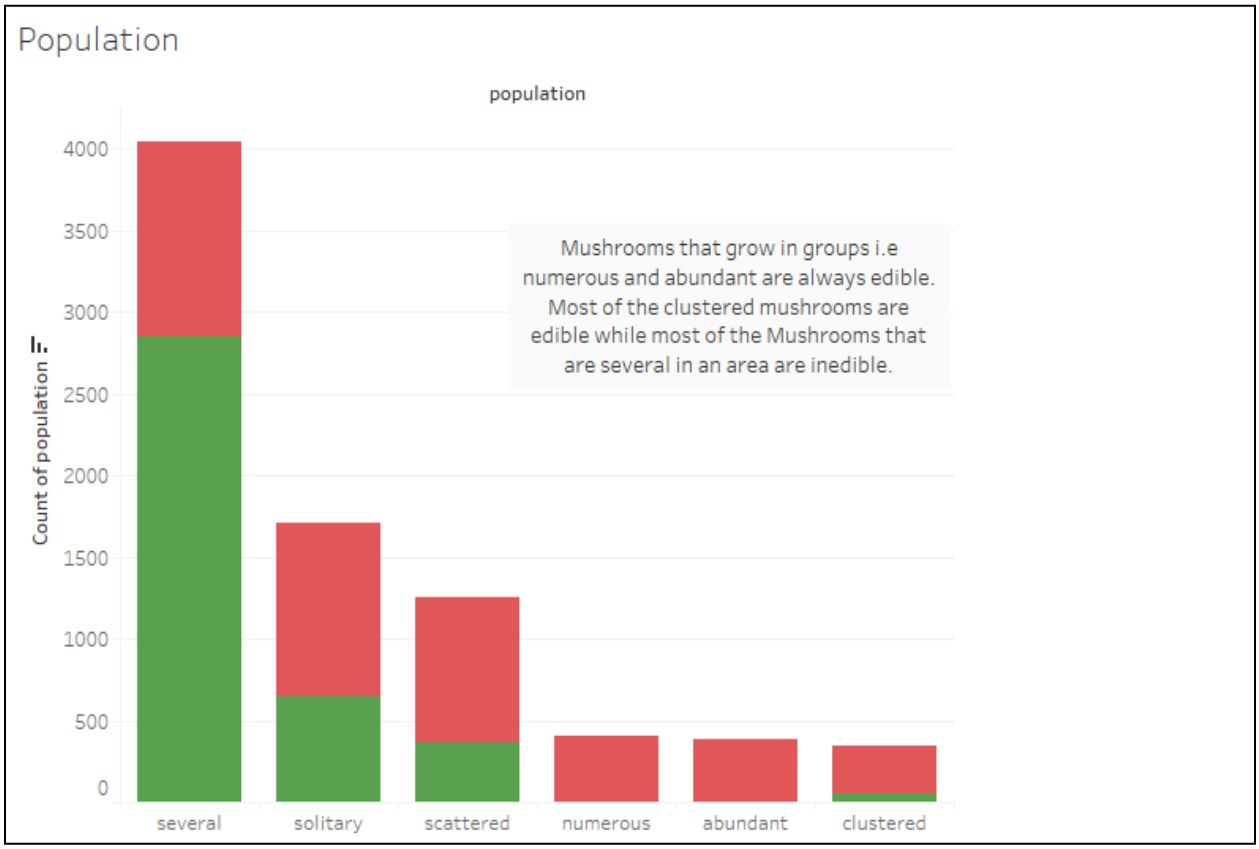
Exploratory Data Analysis:

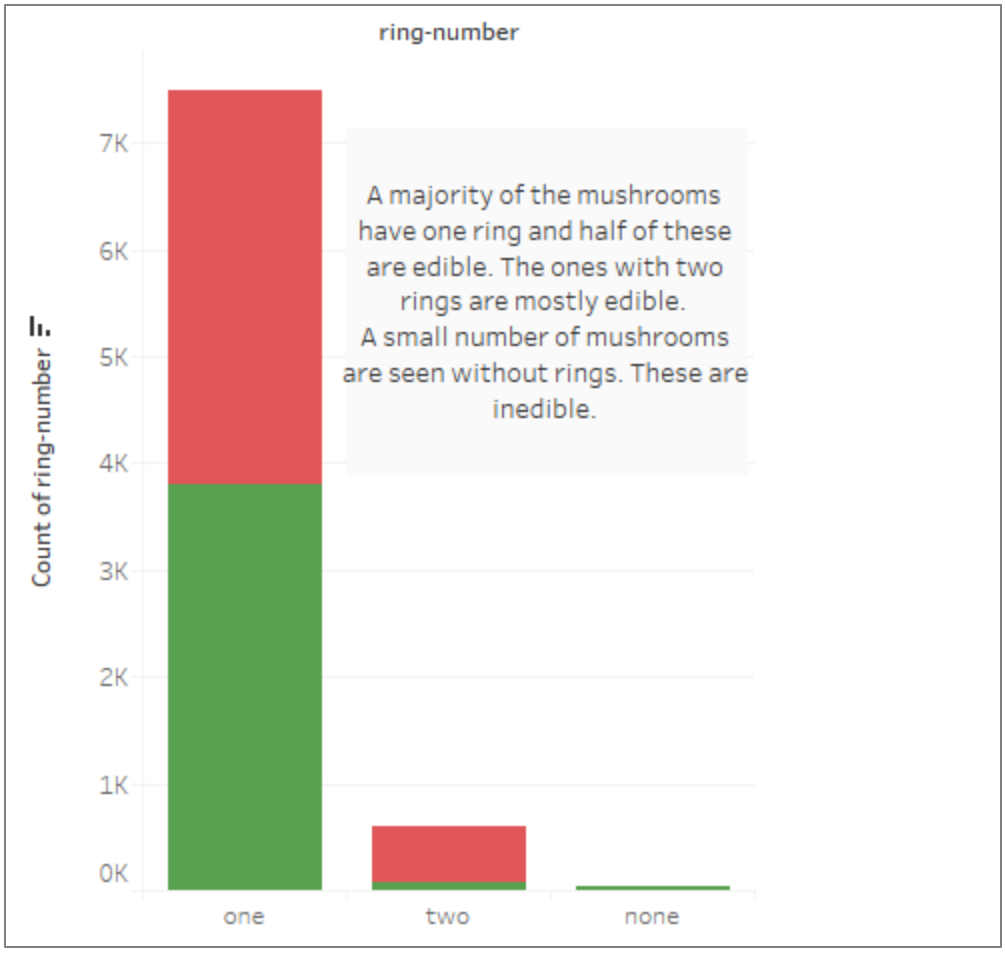
Now let's find out how each of the features is impacting the edibility of mushrooms.

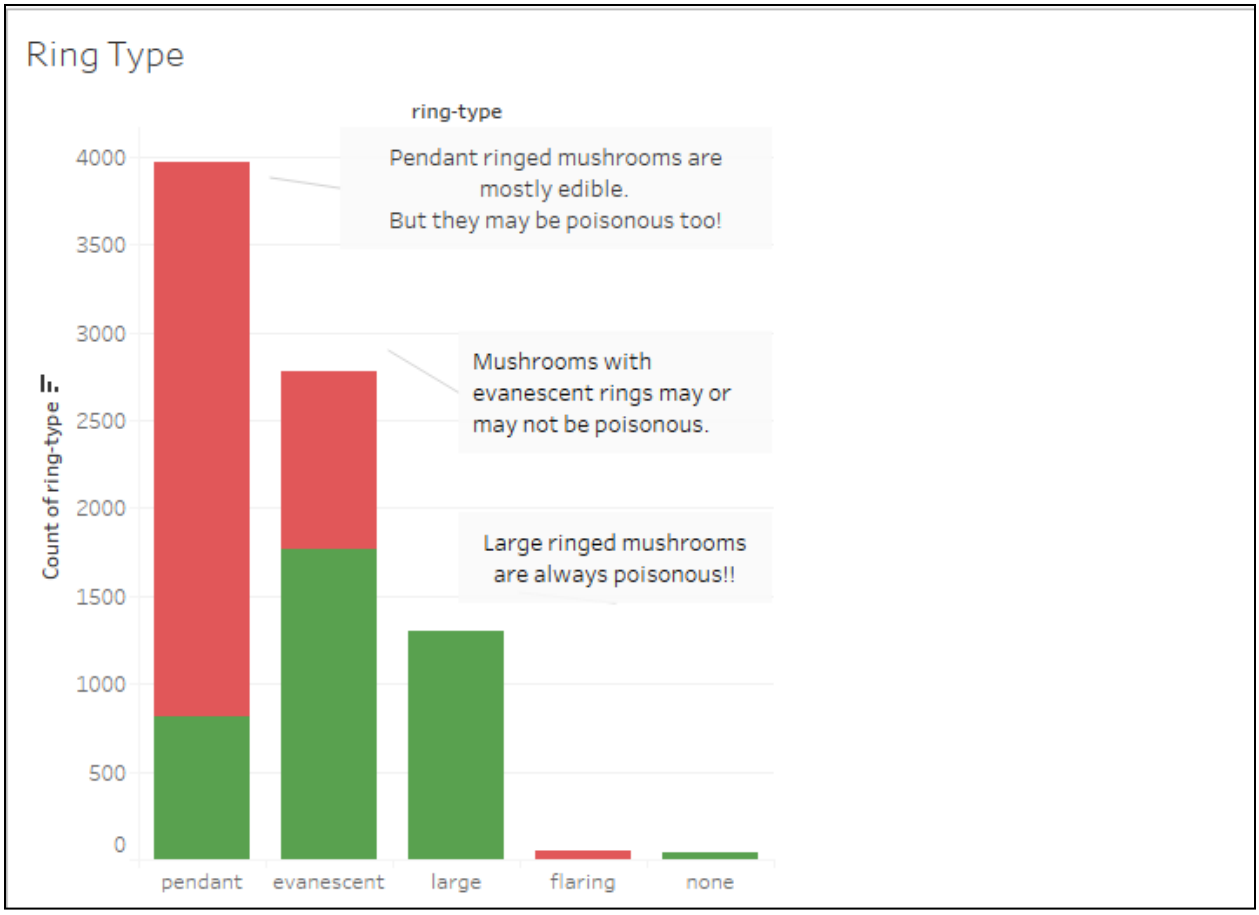
For this, stacked bar graphs have been plotted for each category on X axis and their numbers on Y axis. The red portions of the bars represent edible volumes and the green portions represent the poisonous volumes.

Legend:

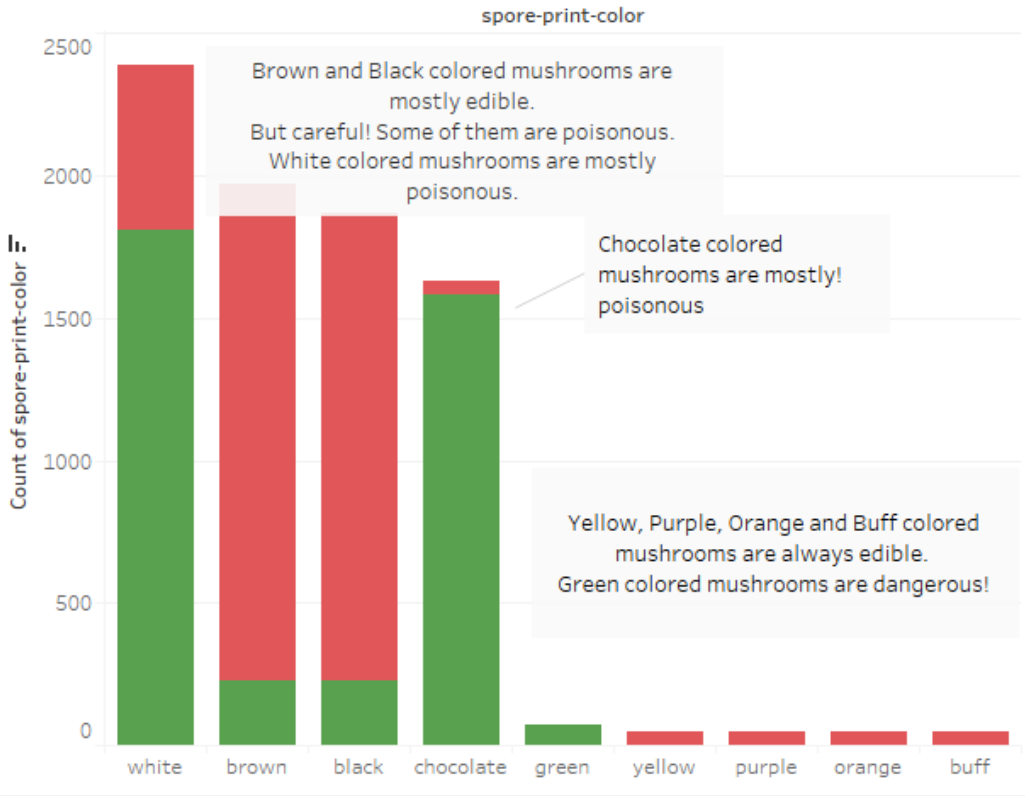




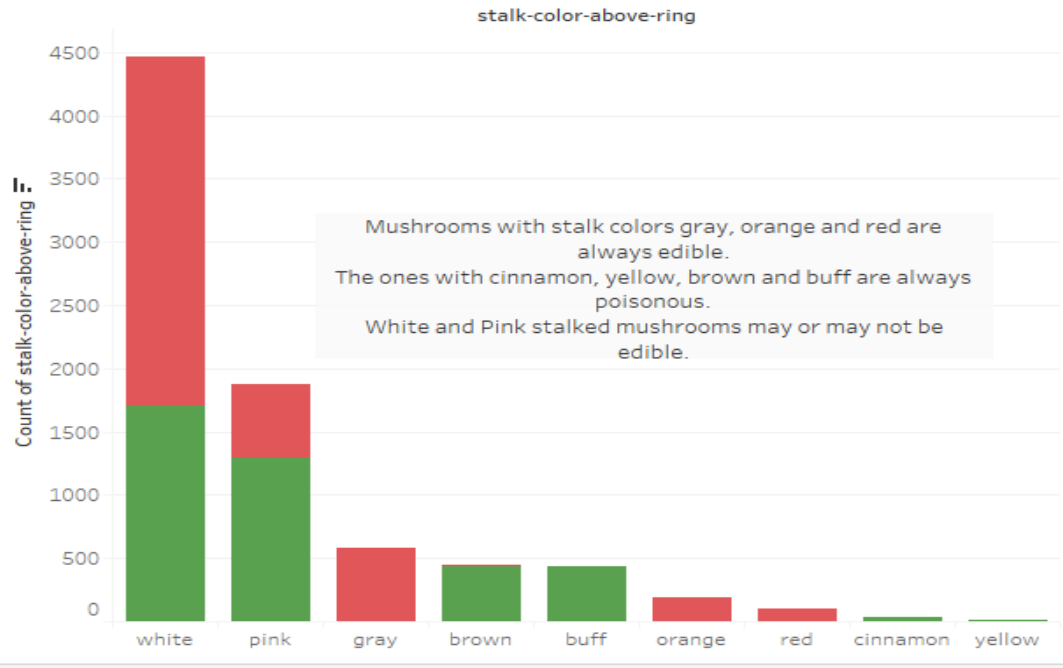

ring-number
Count of ring-number
7K
6K
5K
4K
3K
2K
1K
0K
one
two
none
A majority of the mushrooms have one ring and half of these are edible. The ones with two rings are mostly edible.
A small number of mushrooms are seen without rings. These are inedible.


Ring Type
ring-type
Count of ring-type
4000
3500
3000
2500
2000
1500
1000
500
0
pendant
evanescent
large
flaring
none
Pendant ringed mushrooms are mostly edible.
But they may be poisonous too!
Mushrooms with evanescent rings may or may not be poisonous.
Large ringed mushrooms are always poisonous!!

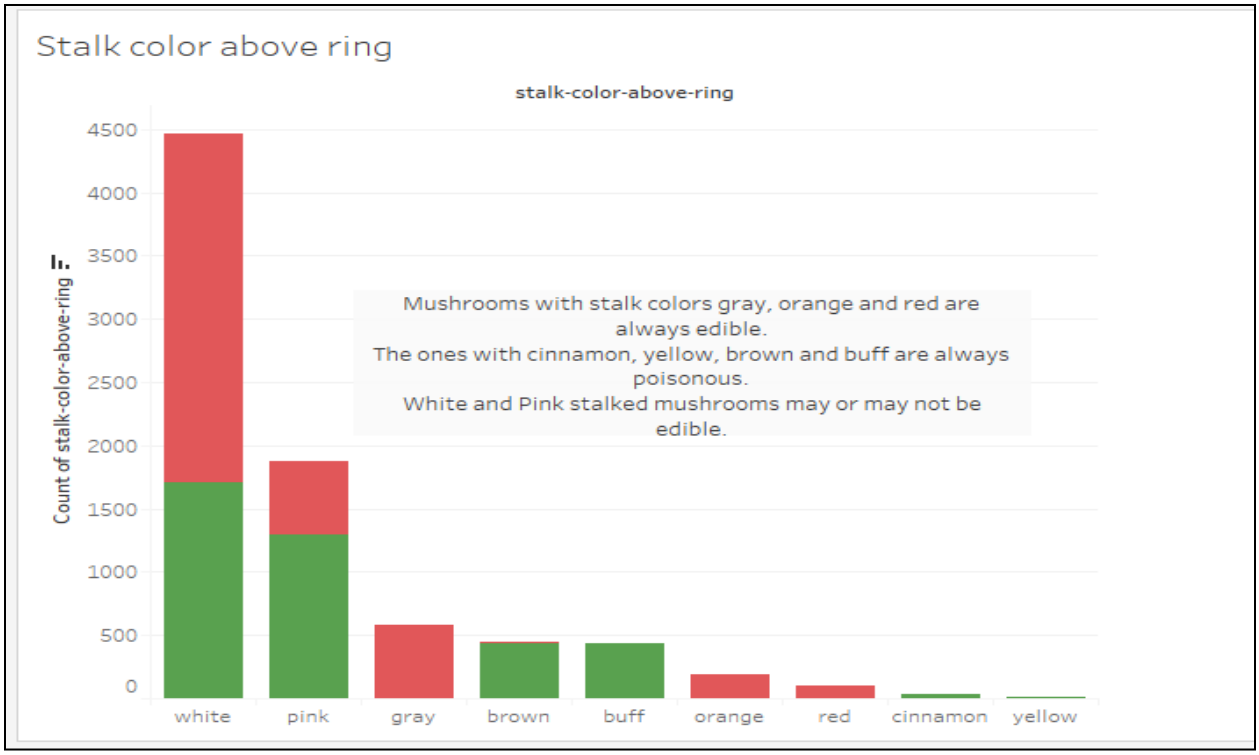
Spore print color



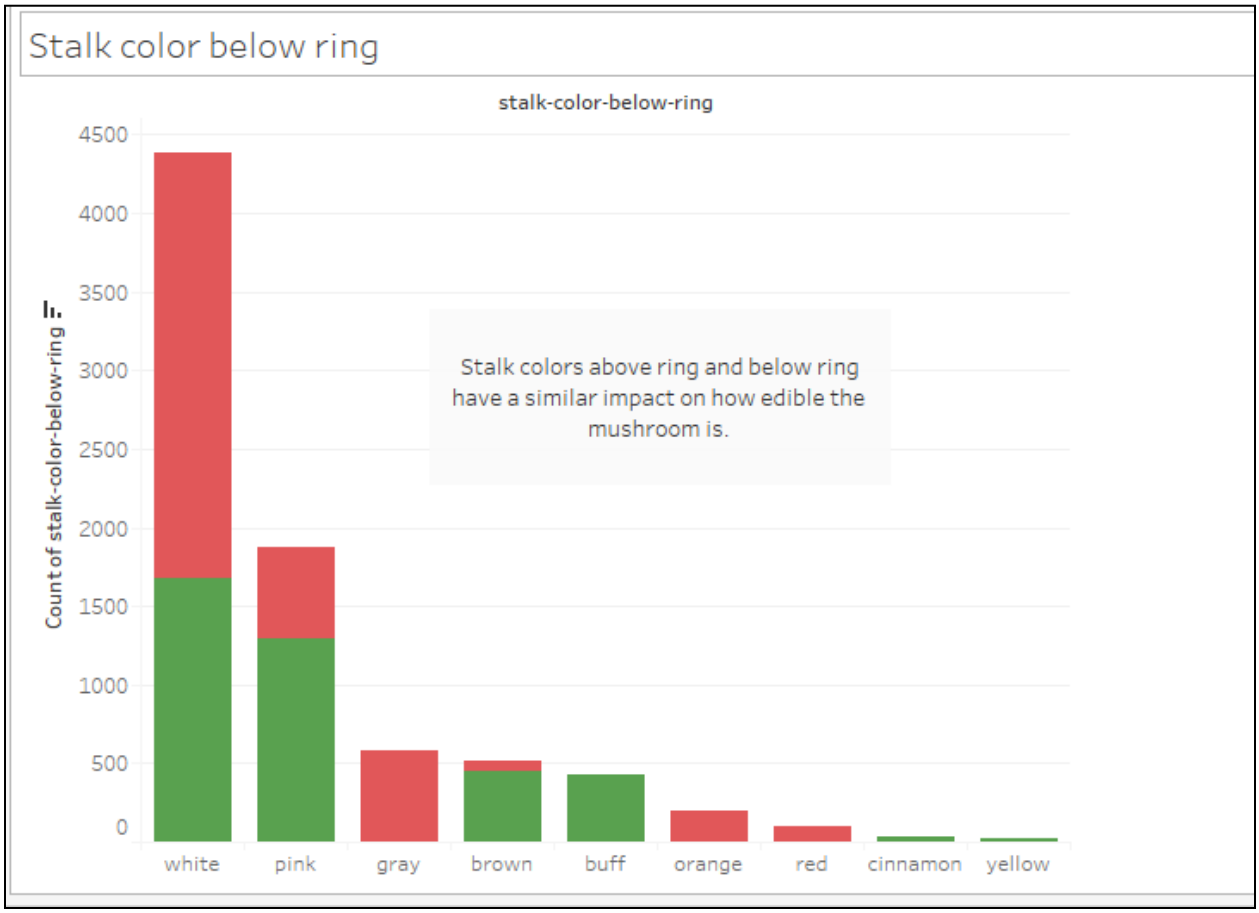
Stalk color above ring

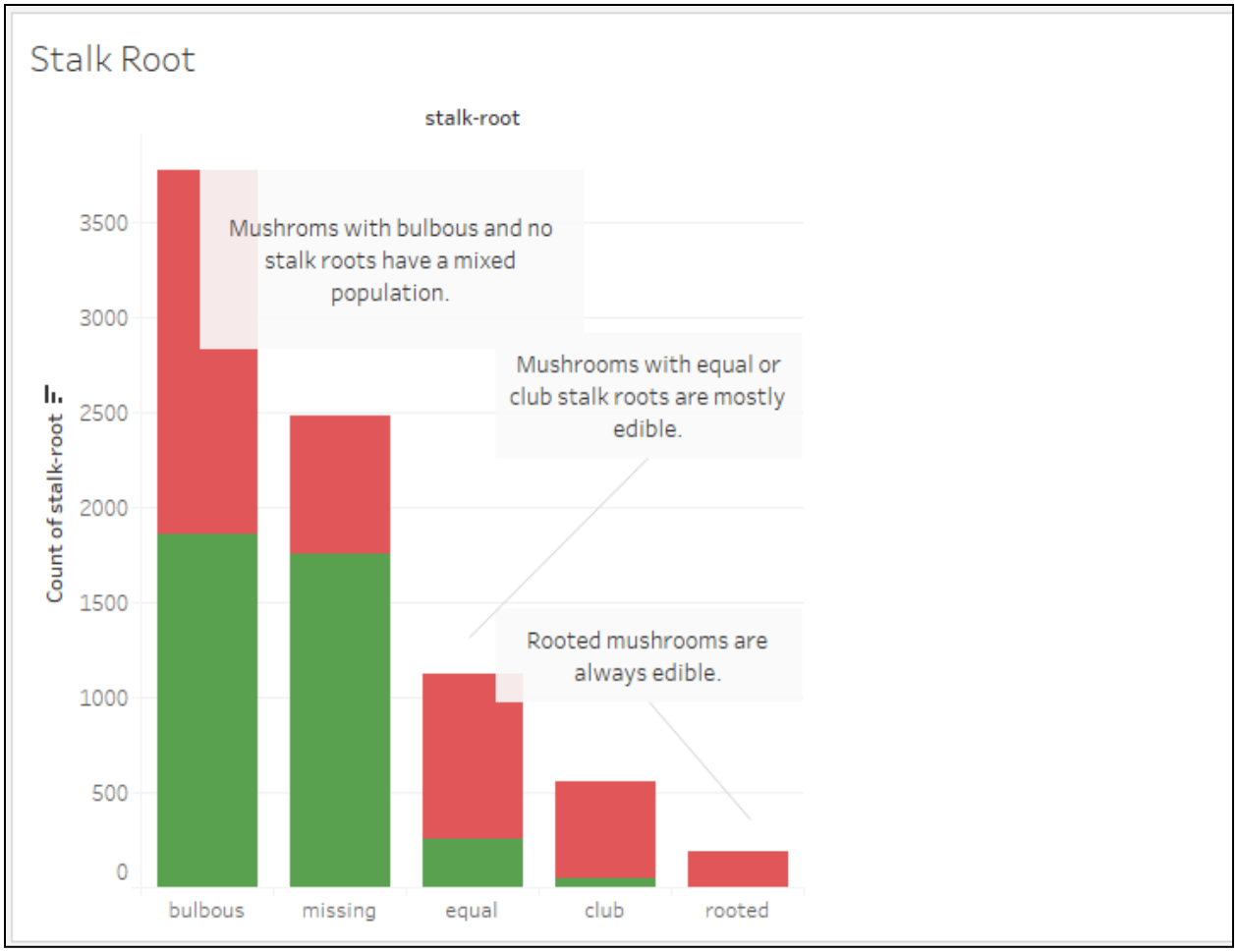


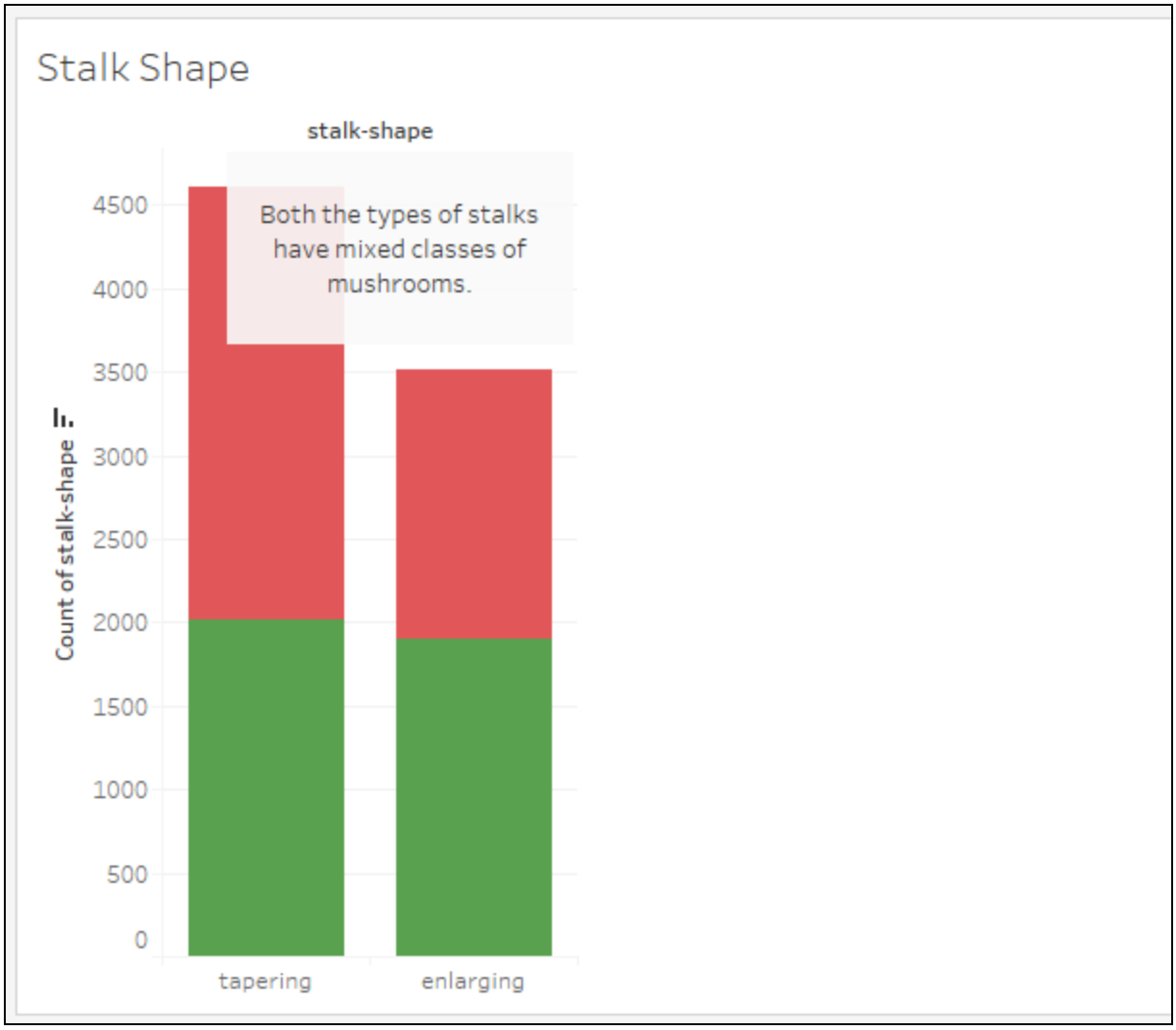
Stalk color above ring

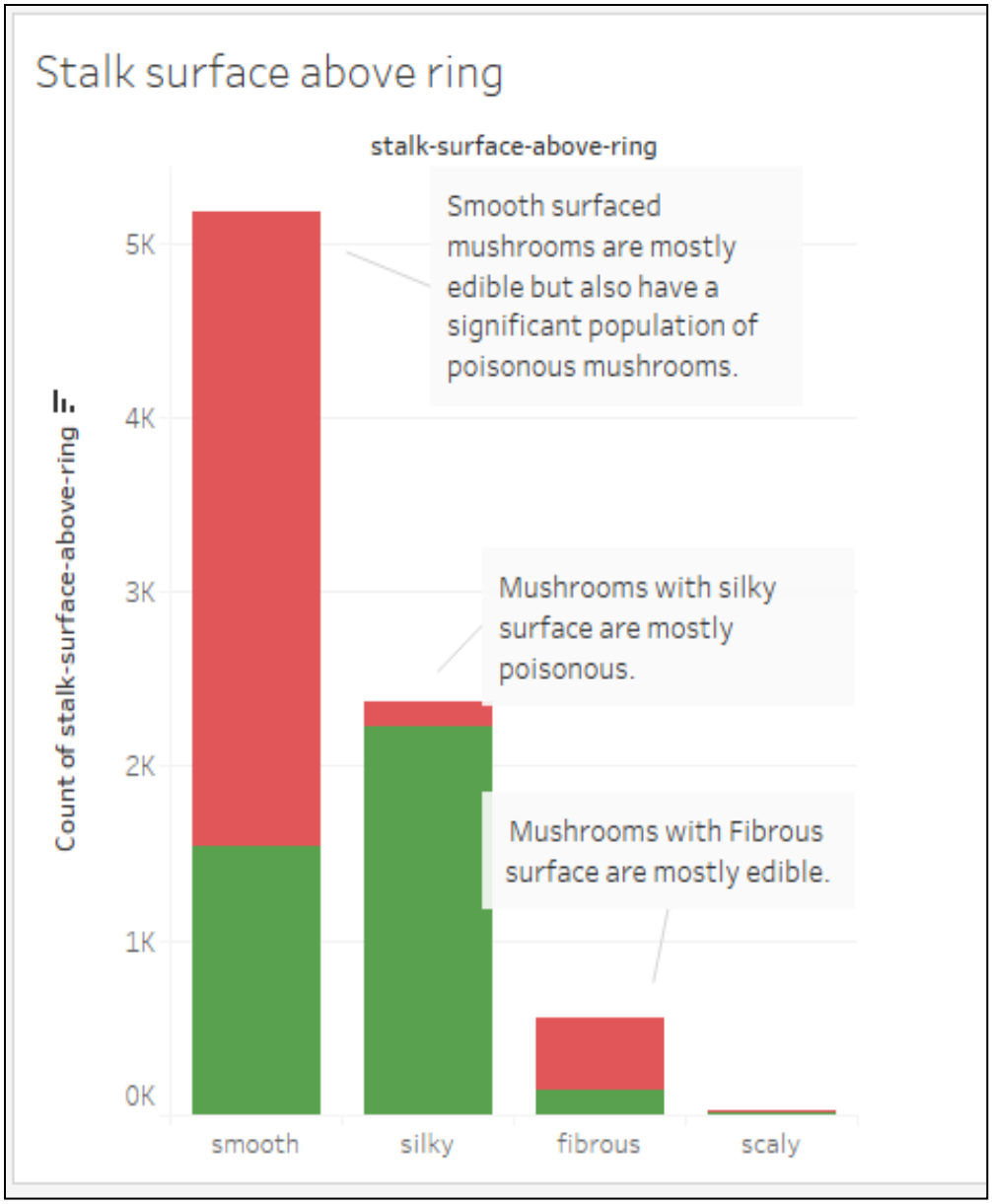


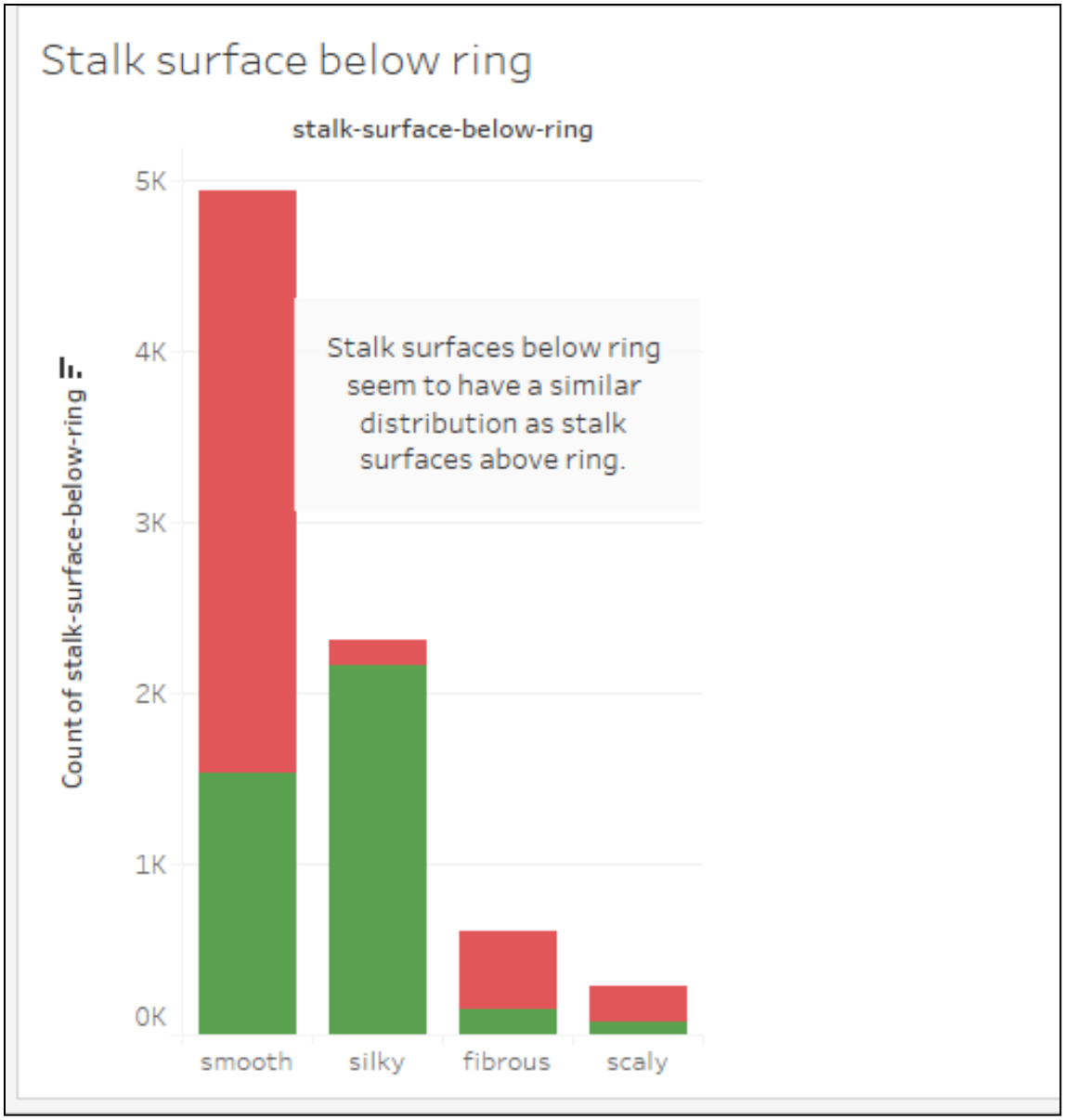
Stalk color below ring

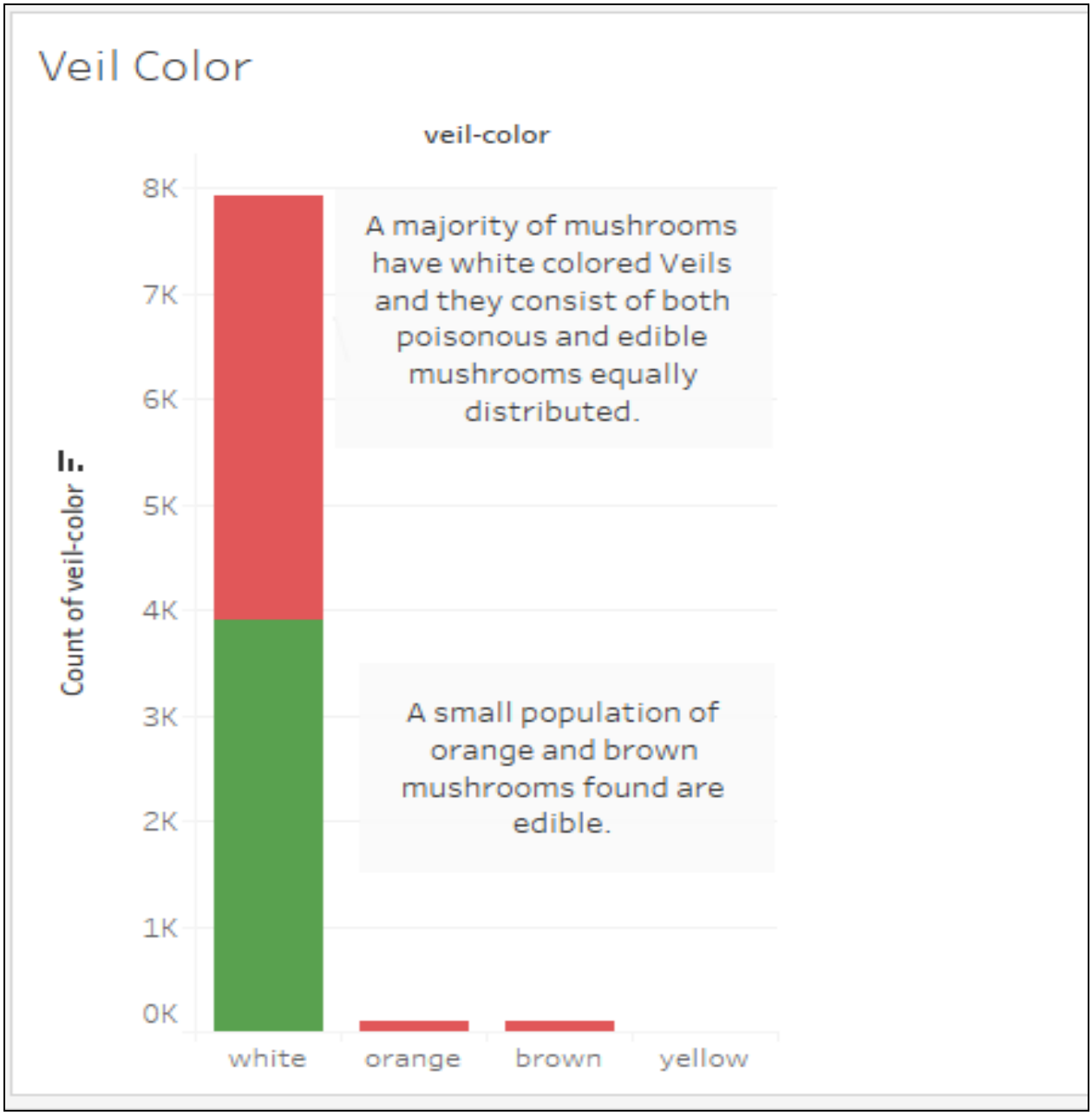



Stalk Root
stalk-root
Count of stalk-root
3500
3000
2500
2000
1500
1000
500
0
Mushroms with bulbous and no stalk roots have a mixed population.
Mushrooms with equal or club stalk roots are mostly edible.
Rooted mushrooms are always edible.
bulbous
missing
equal
club
rooted


Stalk Shape
stalk-shape
Both the types of stalks have mixed classes of mushrooms.
Count of stalk-shape
4500
4000
3500
3000
2500
2000
1500
1000
500
0
tapering
enlarging


Stalk surface above ring
stalk-surface-above-ring
Count of stalk-surface-above-ring
5K
4K
3K
2K
1K
0K
Smooth surfaced mushrooms are mostly edible but also have a significant population of poisonous mushrooms.
Mushrooms with silky surface are mostly poisonous.
Mushrooms with Fibrous surface are mostly edible.
smooth
silky
fibrous
scaly


Stalk surface below ring
stalk-surface-below-ring
Count of stalk-surface-below-ring
5K
4K
3K
2K
1K
0K
Stalk surfaces below ring seem to have a similar distribution as stalk surfaces above ring.
smooth
silky
fibrous
scaly



It's seen that features like Stalk Color, Spore Print Color, Odor and Population Type may have a direct impact on the edibility of mushrooms.

These will be explored further in the modeling step.

Pre-processing:

It is important to process that data so that the models find it easier to read and interpret them. This will also have an impact on the performance of the model.

Since we are dealing will all categorical variables, we will have to encode them before training the models.

Let's find the number of features we have currently and analyze how it might change after encoding.

Finding the number of unique values under each feature.

```
mushroom_data.nunique()
```

```
Unnamed: 0                8124
class                        2
cap-shape                    6
cap-surface                  4
cap-color                   10
bruises                      2
odor                         9
gill-attachment              2
gill-spacing                 2
gill-size                    2
gill-color                  12
stalk-shape                  2
stalk-root                   5
stalk-surface-above-ring     4
stalk-surface-below-ring     4
stalk-color-above-ring       9
stalk-color-below-ring       9
veil-type                    1
veil-color                   4
ring-number                  3
ring-type                    5
spore-print-color            9
population                   6
habitat                      7
```

We will have 119 features (excluding the Index as Unnamed:0) after encoding as per the unique values obtained.

```
#One hot encoding done as all variables are categorical
mushroom_renamed_encoded = pd.get_dummies(mushroom_data)
```

```
mushroom_renamed_encoded.head()
```

	Unnamed: 0	class_edible	class_poisonous	cap-shape_bell	cap-shape_conical	cap-shape_convex	cap-shape_flat	cap-shape_knobbed	cap-shape_sunken	cap-surface_fibrous
0	0	0	1	0	0	1	0	0	0	0
1	1	1	0	0	0	1	0	0	0	0
2	2	1	0	1	0	0	0	0	0	0
3	3	0	1	0	0	1	0	0	0	0
4	4	1	0	0	0	1	0	0	0	0

5 rows × 120 columns

We have got 120 columns. One of them is the Unnamed: 0 column which will be dropped.

After encoding we see a number of repetitive columns. Class_edible and class_poisonous are complementary to each other. Hence we can drop one of these columns.

We will be dropping class_edible and focus only on identifying the poisonous mushrooms.

We also see the stalk_root_missing column and it can be dropped.

The veil type is seen to be partial in all cases hence we need not keep it for further analysis.

```
mushroom_renamed_encoded.shape
```

```
(8124, 120)
```

```
mushroom_renamed_encoded.drop(['Unnamed: 0',
                               'class_edible',
                               'stalk-root_missing',
                               'veil-type_partial'], axis =1, inplace =True)
```

```
mushroom_renamed_encoded.shape
```

```
(8124, 116)
```

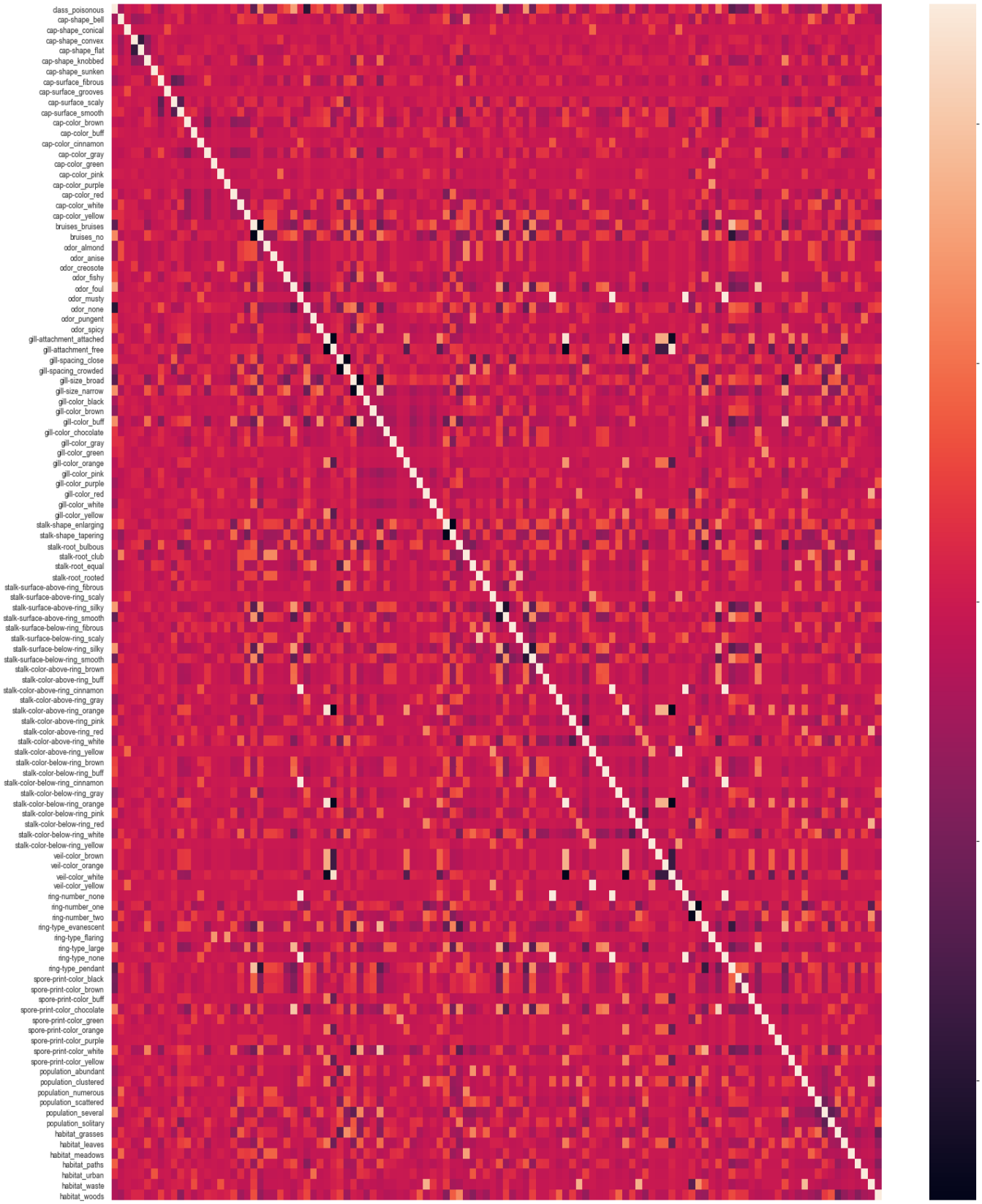
The unnecessary features have been dropped and we are left with 116 features for further analysis.

Correlation Analysis:

The features were analyzed to understand how strong they were related to each other and the outcome.

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(25,25))
ax =plt.subplot(111)
sns.heatmap(mushroom_renamed_encoded.corr(),ax=ax)
```

The heatmap obtained is summarized below:


class_poisonous
cap-shape_bell
cap-shape_conical
cap-shape_convex
cap-shape_flat
cap-shape_knobbed
cap-shape_sunken
cap-surface_fibrous
cap-surface_grooves
cap-surface_scaly
cap-surface_smooth
cap-color_brown
cap-color_buff
cap-color_cinnamon
cap-color_gray
cap-color_green
cap-color_pink
cap-color_purple
cap-color_red
cap-color_white
cap-color_yellow
bruises_bruises
bruises_no
odor_almond
odor_anise
odor_creosote
odor_fishy
odor_foul
odor_musty
odor_none
odor_pungent
odor_spicy
gill-attachment_attached
gill-attachment_free
gill-spacing_close
gill-spacing_crowded
gill-size_broad
gill-size_narrow
gill-color_black
gill-color_brown
gill-color_buff
gill-color_chocolate
gill-color_gray
gill-color_green
gill-color_orange
gill-color_pink
gill-color_purple
gill-color_red
gill-color_white
gill-color_yellow
stalk-shape_enlarging
stalk-shape_tapering
stalk-root_bulbous
stalk-root_club
stalk-root_equal
stalk-root_rooted
stalk-surface-above-ring_fibrous
stalk-surface-above-ring_scaly
stalk-surface-above-ring_silky
stalk-surface-above-ring_smooth
stalk-surface-below-ring_fibrous
stalk-surface-below-ring_scaly
stalk-surface-below-ring_silky
stalk-surface-below-ring_smooth
stalk-color-above-ring_brown
stalk-color-above-ring_buff
stalk-color-above-ring_cinnamon
stalk-color-above-ring_gray
stalk-color-above-ring_orange
stalk-color-above-ring_pink
stalk-color-above-ring_red
stalk-color-above-ring_white
stalk-color-above-ring_yellow
stalk-color-below-ring_brown
stalk-color-below-ring_buff
stalk-color-below-ring_cinnamon
stalk-color-below-ring_gray
stalk-color-below-ring_orange
stalk-color-below-ring_pink
stalk-color-below-ring_red
stalk-color-below-ring_white
stalk-color-below-ring_yellow
veil-color_brown
veil-color_orange
veil-color_white
veil-color_yellow
ring-number_none
ring-number_one
ring-number_two
ring-type_evanescent
ring-type_flaring
ring-type_large
ring-type_none
ring-type_pendant
spore-print-color_black
spore-print-color_brown
spore-print-color_buff
spore-print-color_chocolate
spore-print-color_green
spore-print-color_orange
spore-print-color_purple
spore-print-color_white
spore-print-color_yellow
population_abundant
population_clustered
population_numerous
population_scattered
population_several
population_solitary
habitat_grasses
habitat_leaves
habitat_meadows
habitat_paths
habitat_urban
habitat_waste
habitat_woods
0.8
0.4
0.0
−0.4
−0.8

As seen from the heatmap, bruises_yes, bruises_no, odor_foul, odor_none, gill_size_broad, gill_size_narrow, gill_color_buff, stalk_surface_above_ring_silky, stalk_surface_Above_ring_smooth, ring_type_pendant, spore_print_color_black, spore_print_color_brown,spore_print_color_chocolate, spore_print_color_white, population_several, habitat_paths are having the highest impact on the outcome. Let's get their correlation coefficients for further use.

```
mushroom_renamed_encoded[['class_poisonous',
                          'bruises_bruises', ]
                         'bruises_no',
                         'odor_foul',
                         'odor_none',
                         'gill-size_broad',
                         'gill-size_narrow',
                         'gill-color_buff',
                         'stalk-surface-above-ring_silky',
                         'stalk-surface-above-ring_smooth',
                         'ring-type_pendant',
                         'spore-print-color_black',
                         'spore-print-color_brown',
                         'spore-print-color_chocolate',
                         'spore-print-color_white',
                         'population_several',
                         'habitat_paths']].corr()
```

Correlation co-efficients

	class_poisonous
class_poisonous	1.000000
bruises_bruises	-0.501530
bruises_no	0.501530
odor_foul	0.623842
odor_none	-0.785557
gill-size_broad	-0.540024
gill-size_narrow	0.540024
gill-color_buff	0.538808
stalk-surface-above-ring_silky	0.587658
stalk-surface-above-ring_smooth	-0.491314
ring-type_pendant	-0.540469
spore-print-color_black	-0.396832
spore-print-color_brown	-0.416645
spore-print-color_chocolate	0.490229
spore-print-color_white	0.357384
population_several	0.443722
habitat_paths	0.323346

Bruises, Odor, Gill size, Gill color, Stalk surface above and below ring, Ring type, Spore print color, Population and Habitat seem to have the highest influence on how edible a mushroom is.

Possible signs of mushroom being poisonous:

several in population,
pathy habitat,
white or chocolate colored spore prints,
silky stalk surface,
narrow gill size,
foul odor,
no bruises.

Possible signs of mushroom being edible:

brown or black spore prints,
pendant type ring,
smooth stalk surface,
broad gill size,
no odor,
bruised.

Now let's split the dataset into train test sets for model building.

```
#doing the train test split
mushroom_train,mushroom_test = train_test_split(mushroom_renamed_encoded, test_size=0.3)
```

```
mushroom_train.shape
```

```
(5686, 116)
```

```
mushroom_test.shape
```

```
(2438, 116)
```

We have kept 70% of the data for training and 30% for test analysis.

Storing them in CSV formats for further model training.

```
mushroom_test.to_csv('mushroom_test.csv', index=False)
mushroom_train.to_csv('mushroom_train.csv', index=False)
```

Training and Testing the models:

We will be training a Logistic Regression model , the simplest classification model in usage. The data will first be trained using default parameters and the performance will be assessed. Based on the requirement, the model will be optimized to improve the performance.

```
rModel = linear_model.LogisticRegression()

model = rModel.fit(train_x, train_y)

y_pred= model.predict(test_x)
```

Let's assess the performance of the model using different performance metrics.

```
print('Logistic Regression Performance')
print('Accuracy: ', accuracy_score(test_y,y_pred))
print('Precision: ',precision_score(test_y,y_pred))
print('Recall: ',recall_score(test_y,y_pred))
print('ROC_AUC_Score: ', roc_auc_score(test_y,y_pred))
```

```
Logistic Regression Performance
Accuracy:  0.9983593109105825
Precision:  1.0
Recall:  0.9965986394557823
ROC_AUC_Score:  0.9982993197278911
```

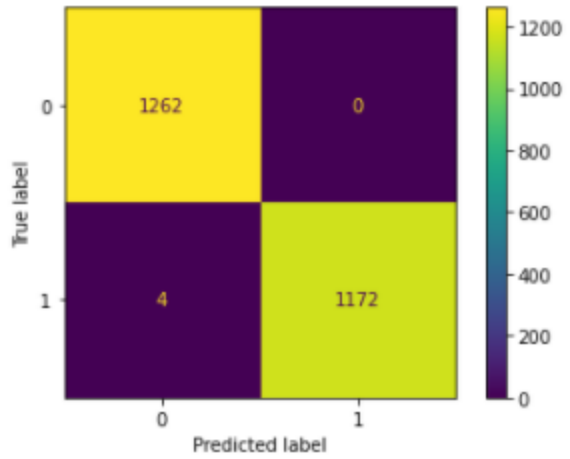
As seen from the above, the model is classifying 99.8% of the cases accurately. Of all the poisonous mushrooms we are able to identify 99.6% (Recall score) and categorize them correctly. The Mushrooms classified as poisonous are actually poisonous in 100% of the cases(Precision) and we have an ROC_AUC of 0.998.

We will be concerned about the Recall more than any other metric. This is because we want the model to correctly identify as many poisonous mushrooms as possible. Else the consequences are fatal.

Let's plot a confusion matrix and have a look at why the Recall is at 99.6%.

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(rModel, test_x, test_y)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatr
```



As seen from the confusion matrix, there are 4 poisonous mushrooms incorrectly identified as edible. 1172 poisonous mushrooms have been correctly identified and so are 1262 edible mushrooms.

Let's try to fit a Random Forest Classifier, one of the most accurate classifiers and understand how the performance fares against logistic regression.

```
RFC =RandomForestClassifier()
```

```
RFC_model = RFC.fit(train_x, train_y)
```

```
y_pred = RFC_model.predict(test_x)
```

Below is the performance of the model with default parameters.

```
print('Random Forest Classifier Performance')
print('Accuracy: ', accuracy_score(test_y,y_pred))
print('Precision: ',precision_score(test_y,y_pred))
print('Recall: ',recall_score(test_y,y_pred))
print('ROC_AUC_Score: ', roc_auc_score(test_y,y_pred))
```

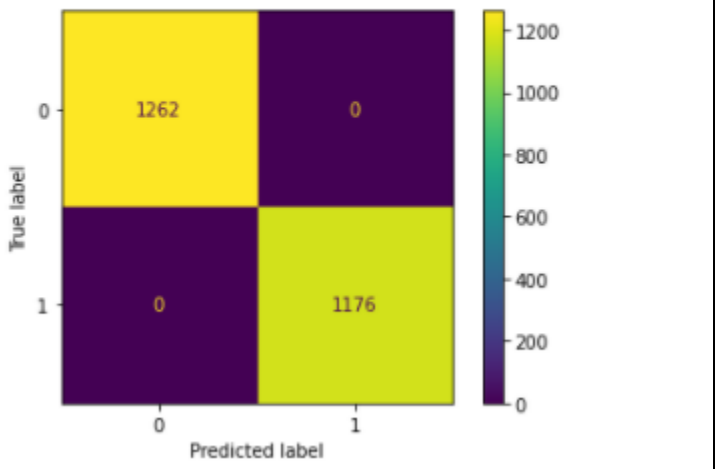
```
Random Forest Classifier Performance
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
ROC_AUC_Score:  1.0
```

We are getting a 100% accuracy and 100% recall. The ROC_AUC_Score is also 1.

Hence the model is classifying all the mushrooms perfectly. Let's plot a confusion matrix and verify the same.

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(RFC, test_x, test_y)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMat
```



There are zero false positives and zero false negatives seen! The models seem to be performing pretty well. Or are they?

Since we are getting a close to 100% accuracy in both the cases, we have to identify a possibility of data leakage. We can also try to fit the models on as minimum features as possible.

Feature Selection:

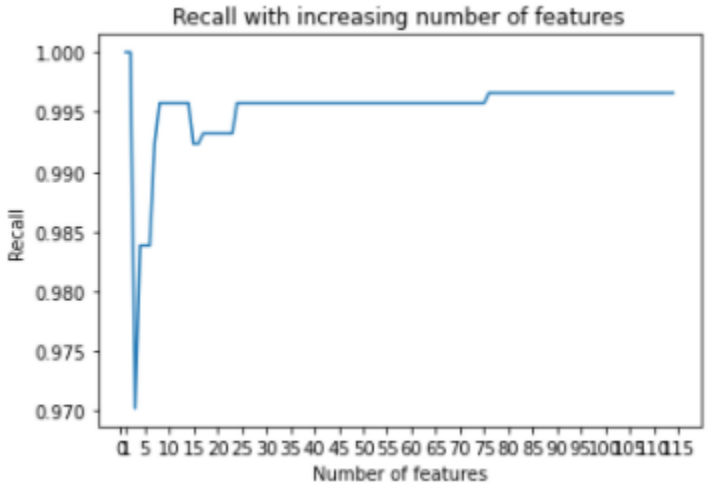
Using the logistic regression model let's try to figure out N-best features. The recall varying with each increase in N will be tracked. This will provide us an understanding of how each feature is impacting the outcome.

We are doing Recursive Feature elimination in order to maintain transparency in the model performance with each N. The method will be looped over 114 times with each value of N and the recall tracked at each stage with the best N features.

```
from sklearn.feature_selection import RFE
recall = []
features =[]
for i in range(1,115):
    rfe =RFE(rModel,i)
    rfe_model = rfe.fit(train_x,train_y)
    columns = np.array(train_x.columns)
    best_columns =[]
    for i in range(train_x.shape[1]):
        if rfe.support_[i] == True:
            |
            best_columns.append(columns[i])
    predicted_y = rModel.fit(train_x[best_columns],train_y).predict(test_x[best_columns])
    score = recall_score(test_y, predicted_y)
    recall.append(score)
    features.append(i)
```

Now, let's plot a graph for the performance of each of these models.

```
plt.plot([i for i in range(1,115)],recall)
plt.title('Recall with increasing number of features')
plt.xlabel('Number of features')
plt.ylabel('Recall')
plt.xticks([0,1,5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,105,110,115])
plt.show()
```



Strangely we see the recall being 1 for the 1 and 2 features and seeing a drastic dip after that. We again see a rise in the recall after 8 features. Not much difference is seen in the model performance after that.

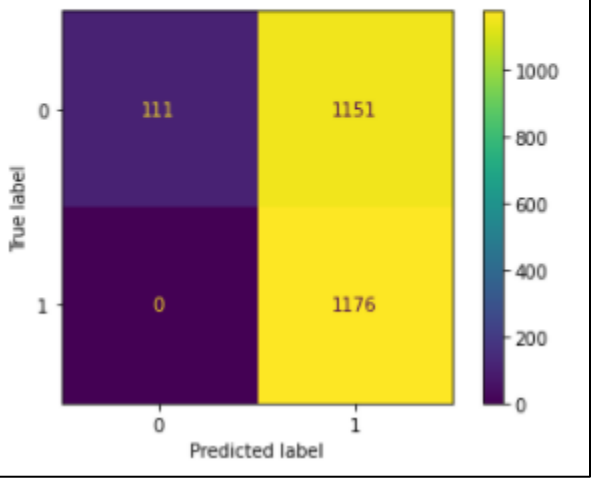
Let's now try to understand which 1 feature can classify all the poisonous mushrooms correctly.

```
from sklearn.feature_selection import RFE
rfe = RFE(rModel,1)
rfe_model = rfe.fit(train_x,train_y)
columns = np.array(train_x.columns)
best_columns =[]
for i in range(train_x.shape[1]):
    if rfe.support_[i] == True:

        best_columns.append(columns[i])
predicted_y = (rModel.fit(train_x[best_columns],train_y)).predict(test_x[best_columns])
print(best_columns, 'is the feature used to train Logistic Regression model')
print('Accuracy Score: ' ,accuracy_score(test_y, predicted_y))
print('Precision: ' ,precision_score(test_y,predicted_y))
print('Recall: ',recall_score(test_y, predicted_y))
print('ROC_AUC_Score: ',roc_auc_score(test_y, predicted_y))
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(rModel, test_x[best_columns], test_y)
```

```
['odor_anise'] is the feature used
Accuracy Score:  0.5278917145200984
Precision:  0.5053717232488182
Recall:  1.0
ROC_AUC_Score:  0.5439778129952457

<sklearn.metrics._plot.confusion_matrix.Cor
```



With Odor_Anise as the only feature, we are getting a recall of 1. However the Accuracy, Precision and ROC_AUC_Score are quite low.

This means that the model is classifying as many mushrooms as poisonous as possible.

Is Odor_Anise a sole indicator for a mushroom being poisonous?

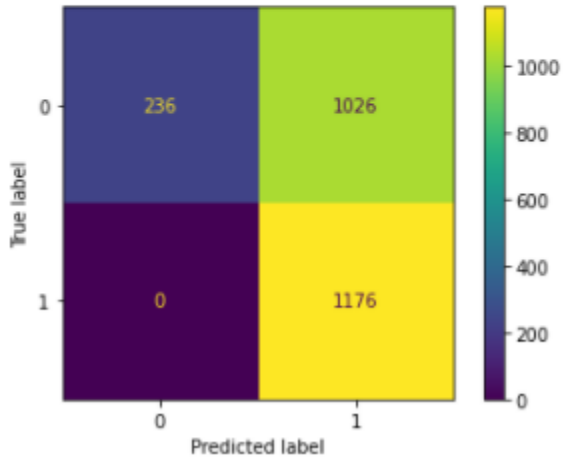
No. As seen in the EDA, all the mushrooms with this odor are edible. Hence our model when trained on this metric only classifies all other mushrooms as poisonous. This model though is able to identify all the poisonous mushrooms correctly; it works at a significant cost of edible mushrooms.

It is also seen that the recall is 1 when 2 best features are used. Let's find out more about which those two features are.

```
rfe = RFE(rModel,2)
rfe_model = rfe.fit(train_x,train_y)
columns = np.array(train_x.columns)
best_columns =[]
for i in range(train_x.shape[1]):
    if rfe.support_[i] == True:
        print(columns[i])
        best_columns.append(columns[i])
predicted_y = (rModel.fit(train_x[best_columns],train_y)).predict(test_x[best_columns])
print('Accuracy: ',accuracy_score(test_y, predicted_y))
print('Precision: ',precision_score(test_y,predicted_y))
print('Recall: ',recall_score(test_y, predicted_y))
print('ROC_AUC_Score', roc_auc_score(test_y, predicted_y))
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(rModel, test_x[best_columns], test_y)
```

```
odor_almond
odor_anise
Accuracy:  0.579163248564397
Precision:  0.5340599455040872
Recall:  1.0
ROC_AUC_Score 0.5935023771790808

<sklearn.metrics._plot.confusion_matrix.Cc
```



It turns out almond odored mushrooms are all edible and hence the model is performing in the same manner as seen with 1 feature.

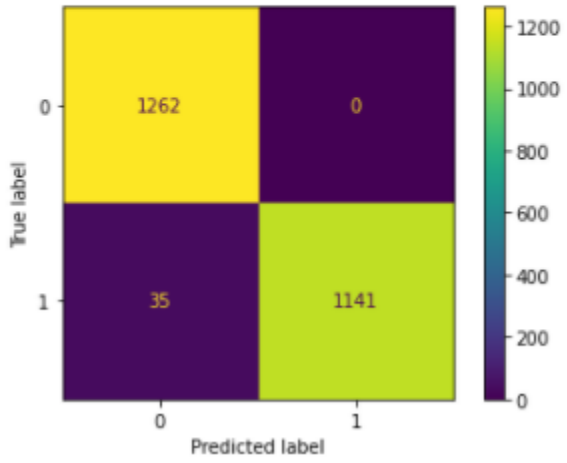
Now what's causing a sudden dip of recall with 3 features?

```
rfe = RFE(rModel,3)
rfe_model = rfe.fit(train_x,train_y)
columns = np.array(train_x.columns)
best_columns =[]
for i in range(train_x.shape[1]):
    if rfe.support_[i] == True:

        best_columns.append(columns[i])
print('The 3 best predictors are: ',best_columns)
predicted_y = (rModel.fit(train_x[best_columns],train_y)).predict(test_x[best_columns])
print('Accuracy: ',accuracy_score(test_y, predicted_y))
print('Precision: ',precision_score(test_y,predicted_y))
print('Recall: ', recall_score(test_y, predicted_y))
print('ROC_AUC_Score: ',roc_auc_score(test_y, predicted_y))
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(rModel, test_x[best_columns], test_y)
```

```
The 3 best predictors are:  ['odor_almond', 'odor_anise', 'odor_none']
Accuracy:  0.9856439704675964
Precision:  1.0
Recall:  0.9702380952380952
ROC_AUC_Score:  0.9851190476190477

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x484
```



Interestingly when three of the odors are introduced the model classifies 35 mushrooms as edible when they are actually poisonous.

This is causing a dip in recall as seen on the graph previously.

However, it is classifying all the edible mushrooms correctly causing a hike in the accuracy.

On the graph it was seen that there was an increase in recall till 8 features and no significant increase seen after that.

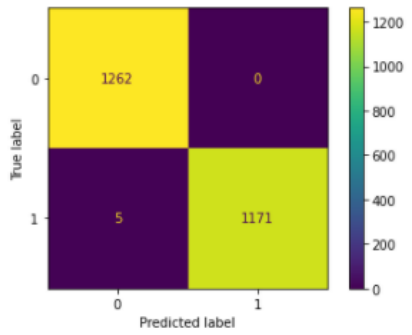
Let’s fit the model on 8 best features and see how the model performs.

```
rfe = RFE(rModel,8)
rfe_model = rfe.fit(train_x,train_y)
columns = np.array(train_x.columns)
best_columns =[]
for i in range(train_x.shape[1]):
    if rfe.support_[i] == True:

        best_columns.append(columns[i])
print('The 8 best predictors are: ',best_columns)
predicted_y = (rModel.fit(train_x[best_columns],train_y)).predict(test_x[best_columns])
print('Accuracy: ',accuracy_score(test_y, predicted_y))
print('Precision: ',precision_score(test_y,predicted_y))
print('Recall: ', recall_score(test_y, predicted_y))
print('ROC_AUC_Score: ',roc_auc_score(test_y, predicted_y))
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(rModel, test_x[best_columns], test_y)
```

```
The 8 best predictors are:  ['odor_almond', 'odor_anise', 'odor_foul', 'odor_none', 'gill-size_narrow', 'stalk-surface-above
-ring_silky', 'stalk-color-below-ring_yellow', 'spore-print-color_green']
Accuracy:  0.9979491386382281
Precision:  1.0
Recall:  0.9957482993197279
ROC_AUC_Score:  0.997874149659864

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x48416f0e10>
```



When 8 features are chosen, the model's performance is close to all the 115 features chosen.

Also the number of misclassified poisonous mushrooms has come down from 35(seen with 5 features) to 5.

Hence we can go ahead with this logistic regression model with 8 features only and get a performance close to usage of 115 features.

We can also see how odor, gill size, stalk surface above ring (silky), yellow stalk color below ring and green spore color are the best indicators of a mushroom being poisonous.

Now that we have seen which features have the highest impact on the outcome, let’s try to optimize the Random Forest model further.

First let’s find out the number of estimators that work the best using GridSearchCV.

```
rfc = RandomForestClassifier()
estimators = [5,7,10,25,50]
gridsearch_rf= GridSearchCV(rfc, param_grid = {'n_estimators': estimators}, cv =5, n_jobs =-1, scoring= 'recall')
```

Have kept the estimators under 100 which is the default value.

```
rfc_model =gridsearch_rf.fit(train_x,train_y)
```

```
print(rfc_model.best_params_,rfc_model.best_score_)
```

```
{'n_estimators': 5} 1.0
```

The model returns a recall of 1 with just 5 estimators!!

Let's find out the maximum number of features required to get a 100% recall.

```
from sklearn.feature_selection import SelectFromModel
sel = SelectFromModel(RandomForestClassifier(n_estimators = 5))
model = sel.fit(train_x, train_y)
best_features = train_x.columns[model.get_support()]
print('The Best features: ' , best_features)
print('Total number of features:', len(best_features))
```

```
The Best features:  Index(['cap-color_buff', 'bruises_no', 'odor_anise', 'odor_none',
       'odor_pungent', 'gill-spacing_crowded', 'gill-size_broad',
       'gill-size_narrow', 'stalk-root_bulbous', 'stalk-root_club',
       'stalk-surface-above-ring_smooth', 'stalk-surface-below-ring_silky',
       'ring-type_pendant', 'spore-print-color_chocolate',
       'spore-print-color_green', 'population_abundant', 'population_several',
       'habitat_paths', 'habitat_urban'],
      dtype='object')
Total number of features: 19
```

We are able to achieve a 100% recall with 19 features only. Odor is again featuring among the top features here.

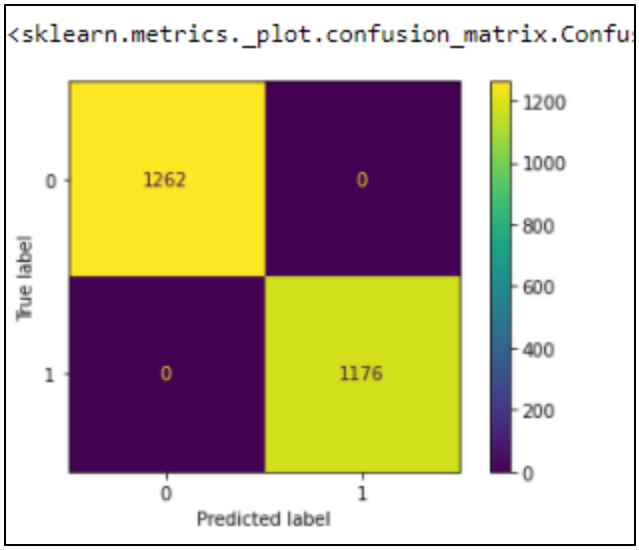
How about other performance estimators? With 5 trees and 19 features, let's figure out.

```
rfc = RandomForestClassifier(n_estimators = 5)
model = rfc.fit(train_x[best_features], train_y)
```

```
y_pred = model.predict(test_x[best_features])
```

```
print('Random Forest Performance with 5 estimators and 19 features')
print('Accuracy: ',accuracy_score(test_y, y_pred))
print('Precision: ',precision_score(test_y,y_pred))
print('Recall: ', recall_score(test_y, y_pred))
print('ROC_AUC_Score: ',roc_auc_score(test_y, y_pred))
plot_confusion_matrix(model, test_x[best_features], test_y)
```

```
Random Forest Performance with 5 estimators and 19 features
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
ROC_AUC_Score:  1.0
```



It can be seen that the model gives a 100% accuracy, precision, recall and an ROC_AUC of 1.

All the mushrooms are correctly classified. This seems to be an amazing model to implement.

Observations and Conclusion:

The **Logistic Regression model** when trained with default parameters gave the following result:

- 4 mushrooms incorrectly classified as edible.
- 1172 poisonous mushrooms correctly identified as poisonous.
- The Precision was seen to be 1 as all mushrooms identified as poisonous were actually poisonous mushrooms.
- The Recall was slightly lower than 100% due to the 4 incorrectly classified mushrooms.

When we tried to apply feature selection on the logistic regression model we obtained the below results:

- Odor was the best indicator for a mushroom being edible or poisonous.
- With 8 features only we could train a model that would give a recall,accuracy and precision close to the one with default parameters.

A **Random Forest Classifier** trained using default parameters gave the below outcomes:

- All mushrooms correctly classified as edible/ poisonous.
- The Precision/Recall and Accuracy were 100% and ROC_AUC was 1.
- Default parameters would mean that 100 trees were used, gini crieterion was applied, no max_depth, at least 2 samples required to split a node and max_features being square root of the number of features(~11 in our case).

We tried to optimize the Random Forest model to run on minimum features and minimum number of trees and below were the results:

- We obtained 100% accuracy/precision and recall with 5 trees.
- The best performance was obtained with 19 features only.
- Odor, Spore color, Stalk size and Color were among the best features identified to obtain the highest accuracy.

It's evident that odor is the main differentiator that can identify a majority of poisonous mushrooms. It can be combined with gill size, stalk surface and spore prints to improve the performance of the models.

It is evident that we go ahead with the Random Forest Classifier model to identify the poisonous mushrooms. This model is picking out all the poisonous mushrooms from the bunch and classifying them correctly.

Yayy! Finally Mario can use the random forest model and identify the deceptive poisonous mushrooms!!!

References:

[1] https://pubmed.ncbi.nlm.nih.gov/30062915/

[2] https://archive.ics.uci.edu/ml/datasets/Mushroom