base model and visualizations

May 16, 2022

[164]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as sm
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score,train_test_split
from sklearn.metrics import mean_squared_error,r2_score,roc_curve,auc
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeRegressor,DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV, ParameterGrid
from sklearn.ensemble import␣
↪BaggingRegressor,BaggingClassifier,RandomForestRegressor,RandomForestClassifier
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
import itertools as it
from sklearn.model_selection import StratifiedKFold, KFold

from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
import time as time
```

[165]:
```
data = pd.read_csv('mushrooms.csv')
```

[166]:
```
data.rename(columns = {'class':'classes'}, inplace=True)
```

[167]:
```
data.columns.to_list()
```

[167]:
```
['classes',
 'cap-shape',
 'cap-surface',
 'cap-color',
 'bruises',
 'odor',
```

```
 'gill-attachment',
 'gill-spacing',
 'gill-size',
 'gill-color',
 'stalk-shape',
 'stalk-root',
 'stalk-surface-above-ring',
 'stalk-surface-below-ring',
 'stalk-color-above-ring',
 'stalk-color-below-ring',
 'veil-type',
 'veil-color',
 'ring-number',
 'ring-type',
 'spore-print-color',
 'population',
 'habitat']
```

[168]:
```
data.columns = data.columns.str.strip().str.lower().str.replace('-', '_')
```

[169]:
```
data.isna().sum()
```

[169]:
```
classes                     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
```

[170]:
```
data.head()
```

[170]:

	classes	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	…

	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

	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

	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

[5 rows x 23 columns]

[171]:
```
classes = {
    'e':'edible',
    'p':'poisonous'
}

cap_shapes = {
    'b':'bell',
    'c':'conical',
    'x':'convex',
    'f':'flat',
    'k':'knobbed',
    's':'sunken'
}

cap_surfaces = {
```

```
    'f':'fibrous',
    'g':'grooves',
    'y':'scaly',
    's':'smooth'
}

cap_colors = {
    'n':'brown',
    'b':'buff',
    'c':'cinnamon',
    'g':'gray',
    'r':'green',
    'p':'pink',
    'u':'purple',
    'e':'red',
    'w':'white',
    'y':'yellow'
}

bruise_class = {
    't':'bruises',
    'f':'no_bruises'
}

odors = {
    'a':'almond',
    'l':'anise',
    'c':'creosote',
    'y':'fishy',
    'f':'foul',
    'm':'musty',
    'n':'none',
    'p':'pungent',
    's':'spicy'
}

gill_attachments = {
    'a':'attached',
    'd':'descending',
    'f':'free',
    'n':'notched'
}

gill_spacings = {
    'c':'close',
    'w':'crowded',
    'd':'distant'
```

```
}

gill_sizes = {
    'b':'broad',
    'n':'narrow'
}

gill_colors = {
    'k':'black',
    'n':'brown',
    'b':'buff',
    'h':'chocolate',
    'g':'gray',
    'r':'green',
    'o':'orange',
    'p':'pink',
    'u':'purple',
    'e':'red',
    'w':'white',
    'y':'yellow'
}

stalk_shapes = {
    'e':'enlarging',
    't':'tapering'
}

stalk_roots = {
    'b':'bulbous',
    'c':'club',
    'u':'cup',
    'e':'equal',
    'z':'rhizomorphs',
    'r':'rooted',
    '?':'NA'
}

stalk_surface_above_rings = {
    'f':'fibrous',
    'y':'scaly',
    'k':'silky',
    's':'smooth'
}

stalk_surface_below_rings = {
    'f':'fibrous',
    'y':'scaly',
```

```
    'k':'silky',
    's':'smooth'
}

stalk_color_above_rings = {
    'n':'brown',
    'b':'buff',
    'c':'cinnamon',
    'g':'gray',
    'o':'orange',
    'p':'pink',
    'e':'red',
    'w':'white',
    'y':'yellow'
}

stalk_color_below_rings = {
    'n':'brown',
    'b':'buff',
    'c':'cinnamon',
    'g':'gray',
    'o':'orange',
    'p':'pink',
    'e':'red',
    'w':'white',
    'y':'yellow'
}

veil_types = {
    'p':'partial',
    'u':'universal'
}

veil_colors = {
    'n':'brown',
    'o':'orange',
    'w':'white',
    'y':'yellow'
}

ring_numbers = {
    'n':'none',
    'o':'one',
    't':'two'
}

ring_types = {
```

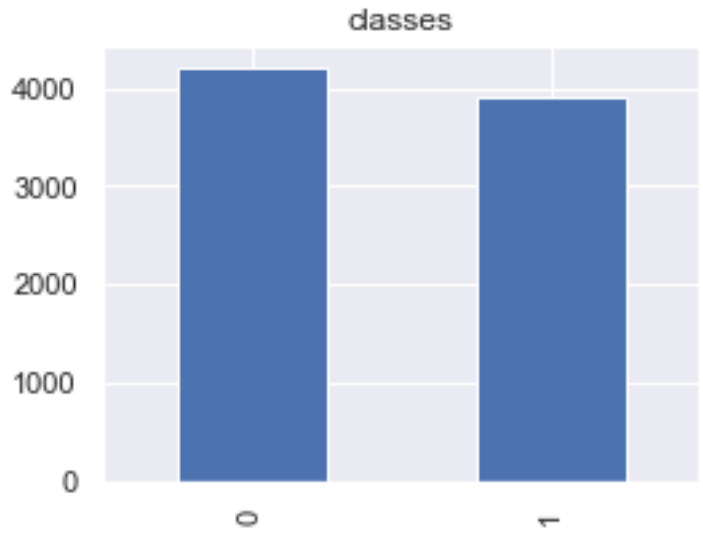
```
    'c':'cobwebby',
    'e':'evanescent',
    'f':'flaring',
    'l':'large',
    'n':'none',
    'p':'pendant',
    's':'sheathing',
    'z':'zone'
}

spore_print_colors = {
    'k':'black',
    'n':'brown',
    'b':'buff',
    'h':'chocolate',
    'r':'green',
    'o':'orange',
    'u':'purple',
    'w':'white',
    'y':'yellow'
}

populations = {
    'a':'abundant',
    'c':'clustered',
    'n':'numerous',
    's':'scattered',
    'v':'several',
    'y':'solitary'
}

habitats = {
    'g':'grasses',
    'l':'leaves',
    'm':'meadows',
    'p':'paths',
    'u':'urban',
    'w':'waste',
    'd':'woods'
}
```

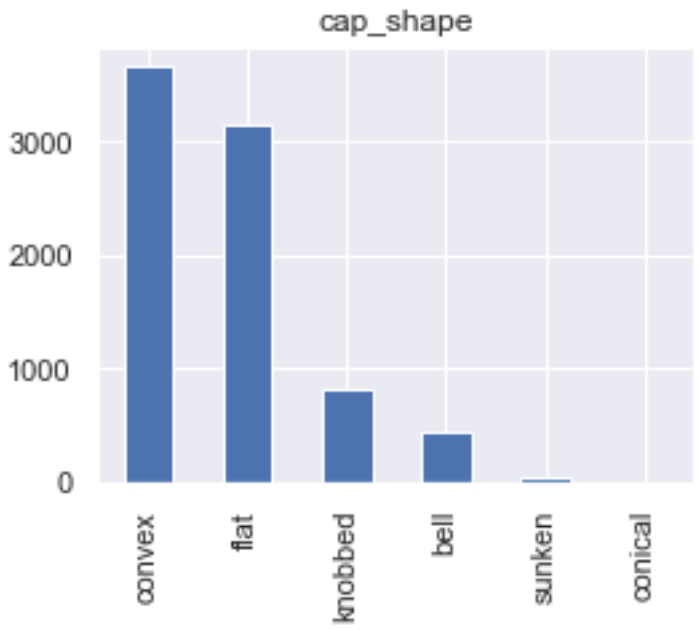
[172]:
```
data.replace({'classes': classes,
              'cap_shape': cap_shapes,
              'cap_surface': cap_surfaces,
              'cap_color': cap_colors,
              'bruises': bruise_class,
              'odor': odors,
```

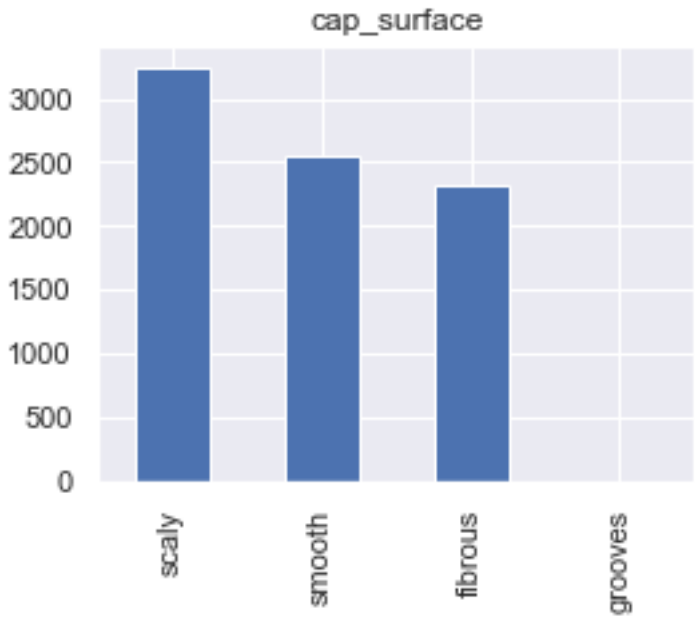
```
                'gill_attachment': gill_attachments,
                'gill_spacing': gill_spacings,
                'gill_size': gill_sizes,
                'gill_color': gill_colors,
                'stalk_shape': stalk_shapes,
                'stalk_root': stalk_roots,
                'stalk_surface_above_ring': stalk_surface_above_rings,
                'stalk_surface_below_ring': stalk_surface_below_rings,
                'stalk_color_above_ring': stalk_color_above_rings,
                'stalk_color_below_ring': stalk_color_below_rings,
                'veil_type': veil_types,
                'veil_color': veil_colors,
                'ring_number': ring_numbers,
                'ring_type': ring_types,
                'spore_print_color': spore_print_colors,
                'population': populations,
                'habitat': habitats},
               inplace=True)
```

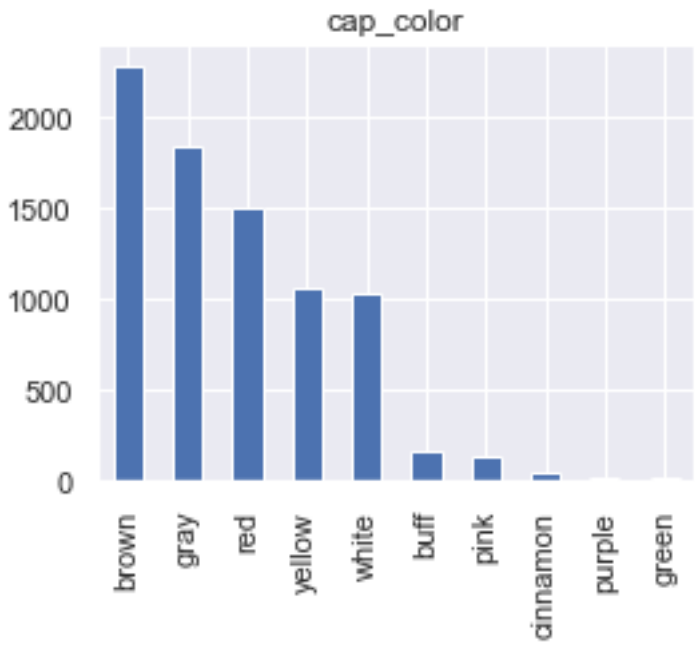
[173]:
```
data['classes'] = np.where(data['classes'] == 'poisonous', 1, 0)
```

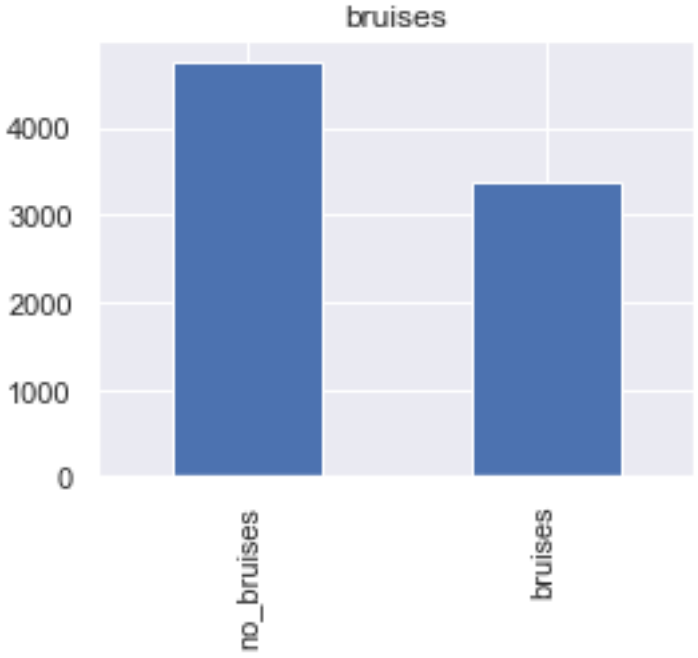
[200]:
```
for col in data.columns:
    plt.figure(figsize = (4,3))
    data[col].value_counts().plot(kind='bar')
    plt.title(col)
    plt.show()
```

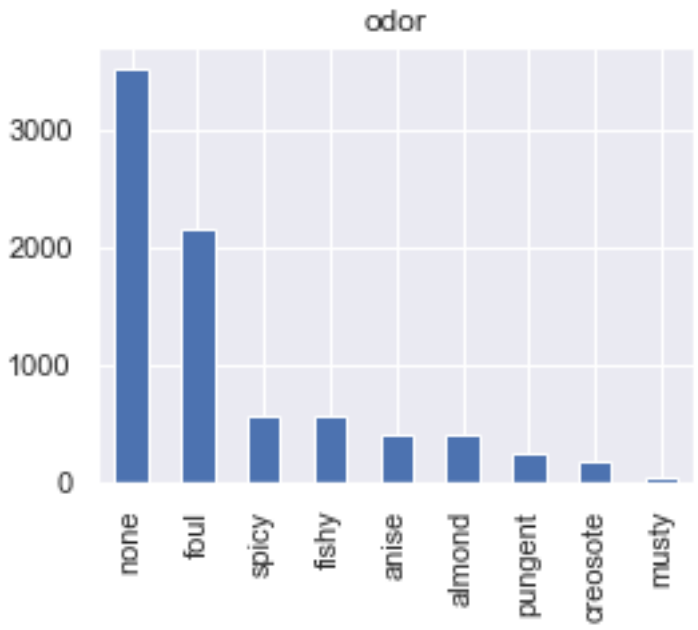


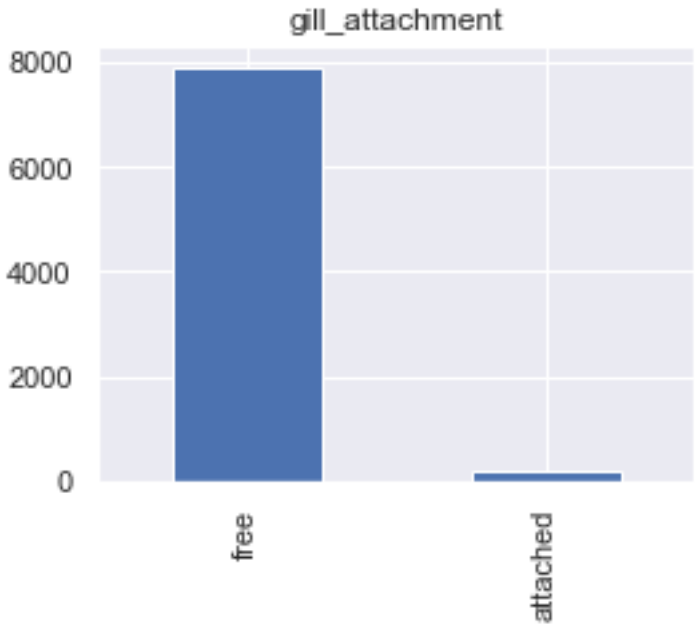

cap_shape
3000
2000
1000
0
convex
flat
knobbed
bell
sunken
conical

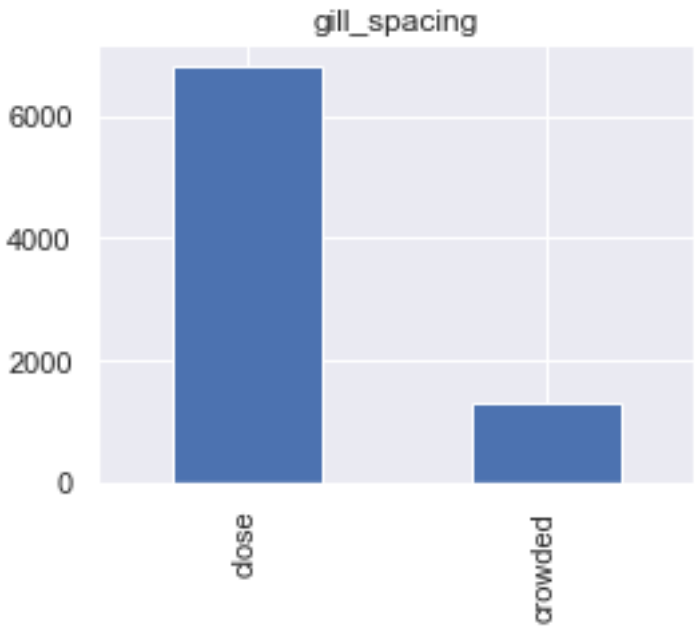

cap_surface
3000
2500
2000
1500
1000
500
0
scaly
smooth
fibrous
grooves

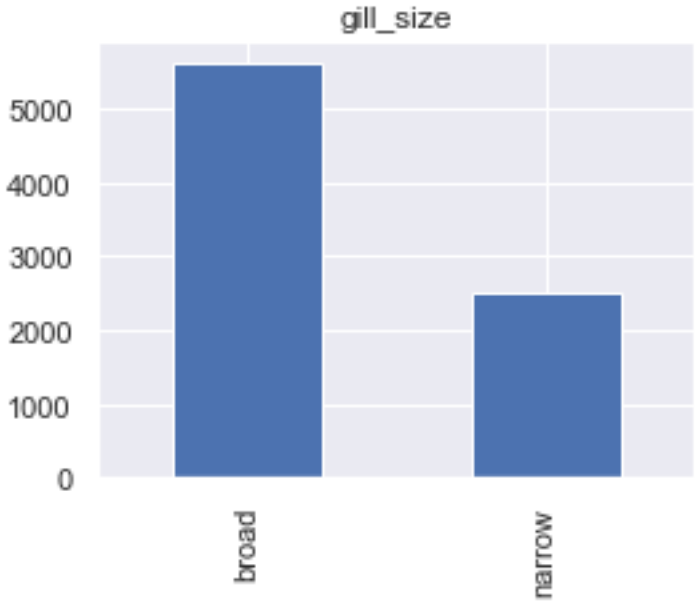

cap_color
2000
1500
1000
500
0
brown
gray
red
yellow
white
buff
pink
cinnamon
purple
green

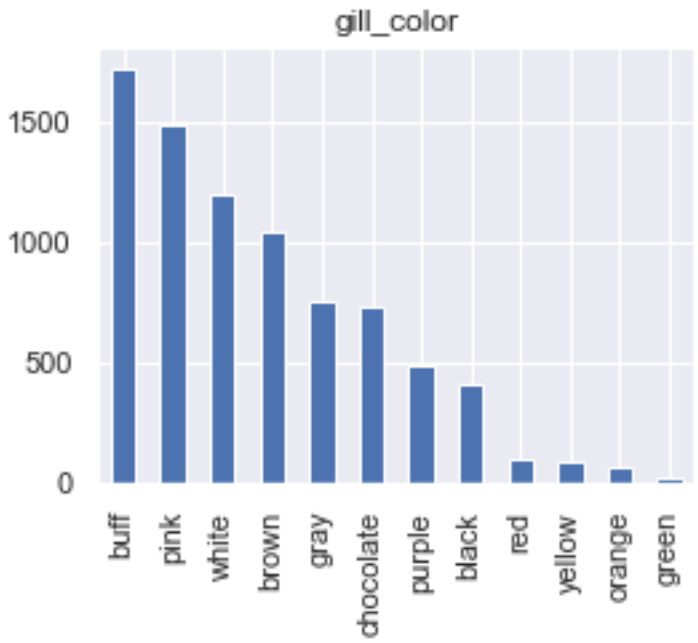

bruises
4000
3000
2000
1000
0
no_bruises
bruises

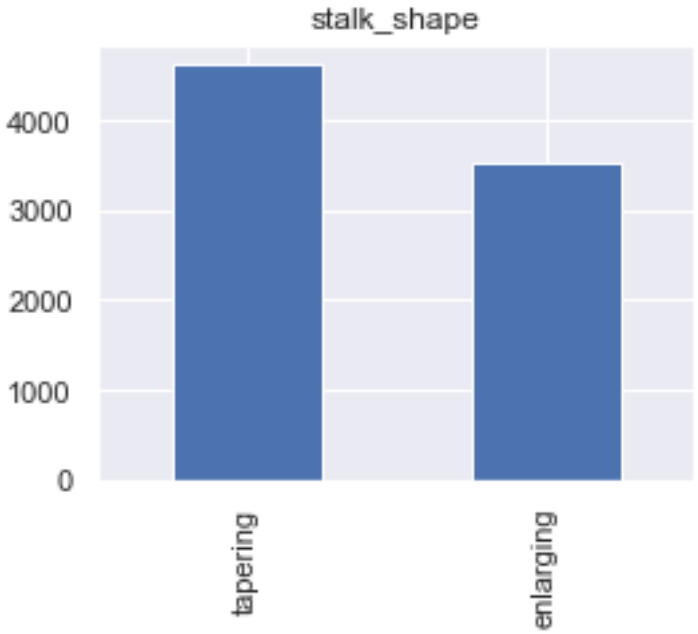

odor
3000
2000
1000
0
none
foul
spicy
fishy
anise
almond
pungent
creosote
musty

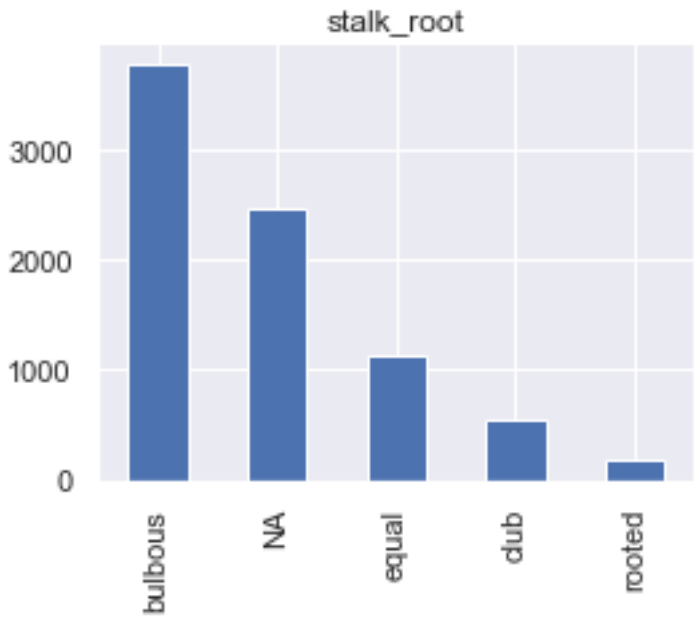

gill_attachment
8000
6000
4000
2000
0
free
attached

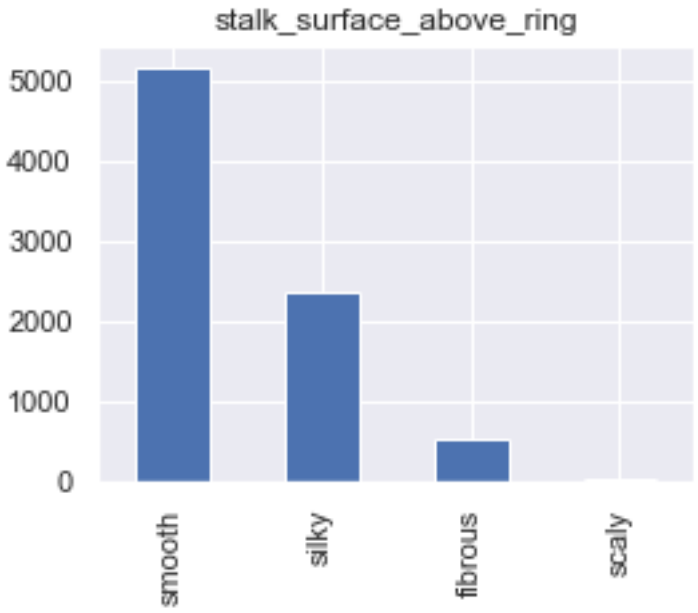

gill_spacing
6000
4000
2000
0
dose
crowded

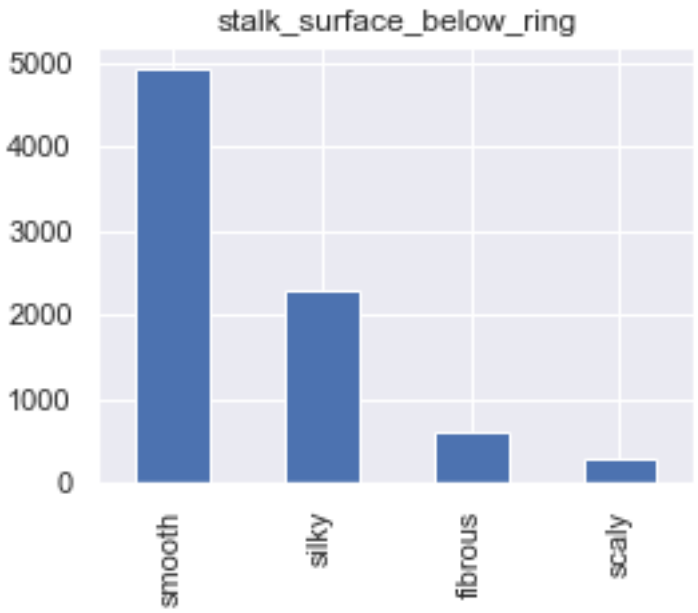

gill_size
5000
4000
3000
2000
1000
0
broad
narrow

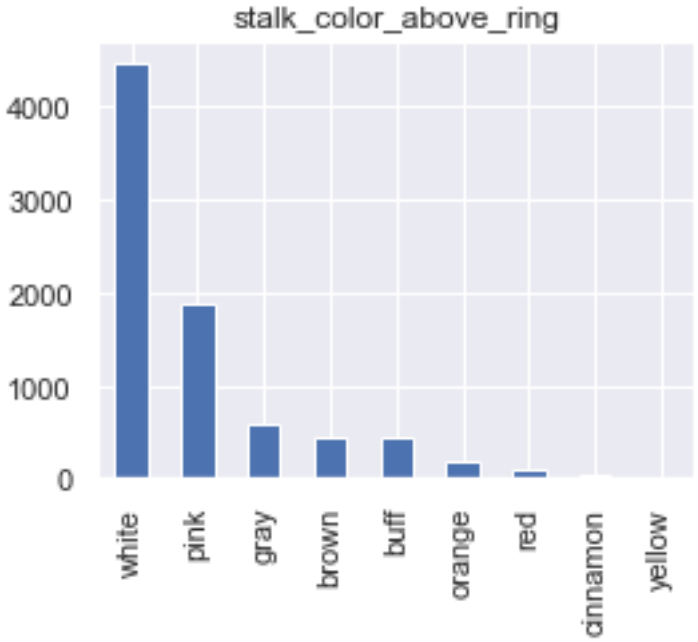

gill_color
1500
1000
500
0
buff
pink
white
brown
gray
chocolate
purple
black
red
yellow
orange
green

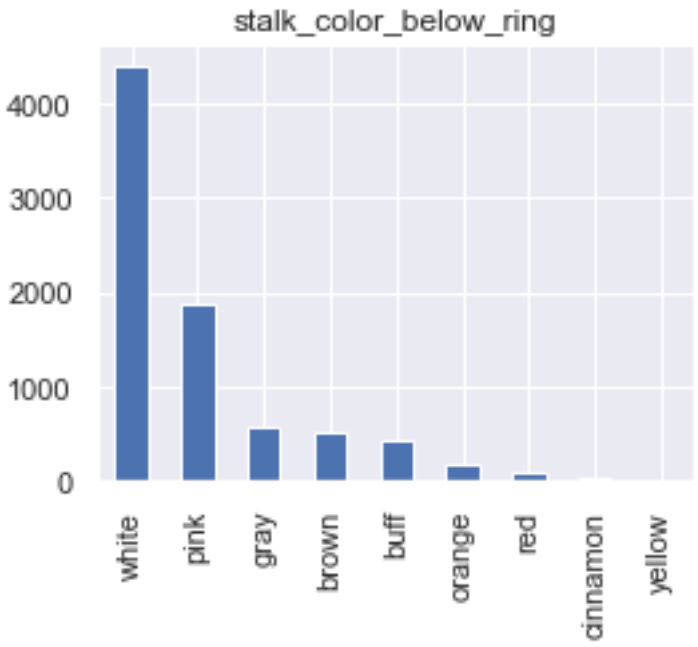

stalk_shape
4000
3000
2000
1000
0
tapering
enlarging

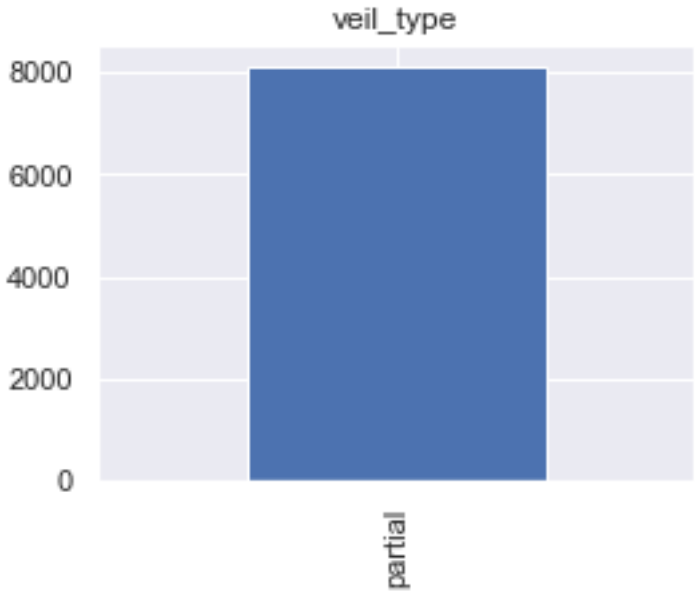

stalk_root
3000
2000
1000
0
bulbous
NA
equal
dub
rooted

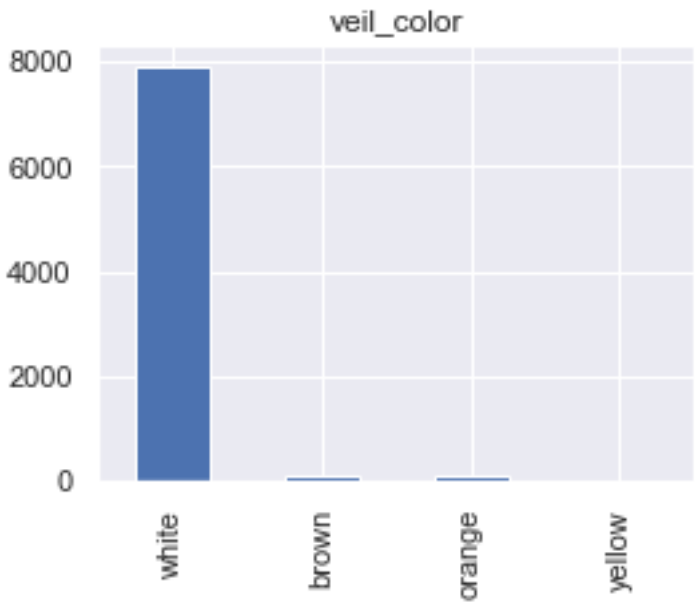

stalk_surface_above_ring
5000
4000
3000
2000
1000
0
smooth
silky
fibrous
scaly

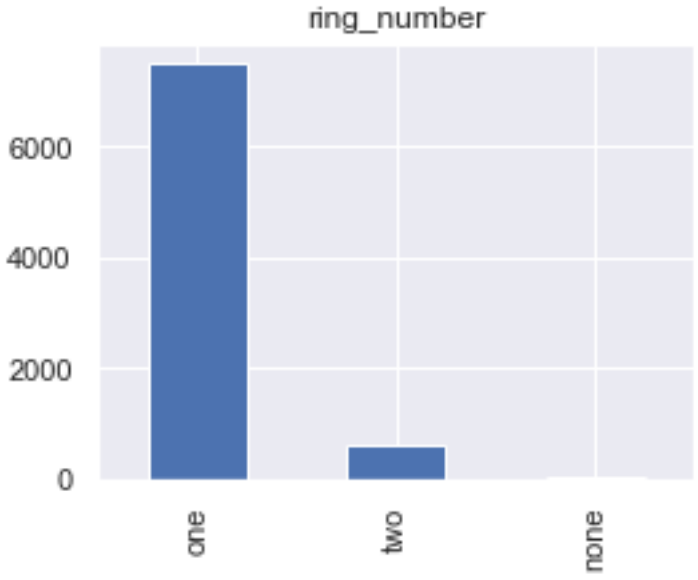

stalk_surface_below_ring
5000
4000
3000
2000
1000
0
smooth
silky
fibrous
scaly

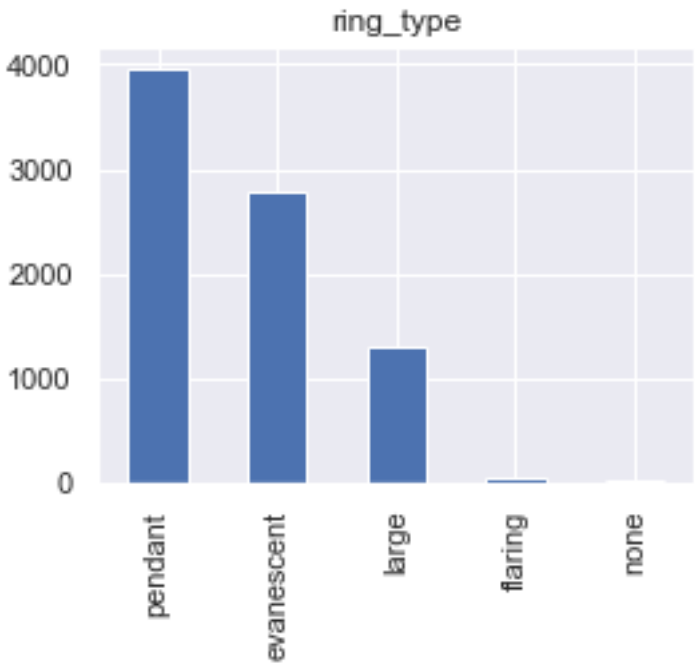

stalk_color_above_ring
4000
3000
2000
1000
0
white
pink
gray
brown
buff
orange
red
cinnamon
yellow

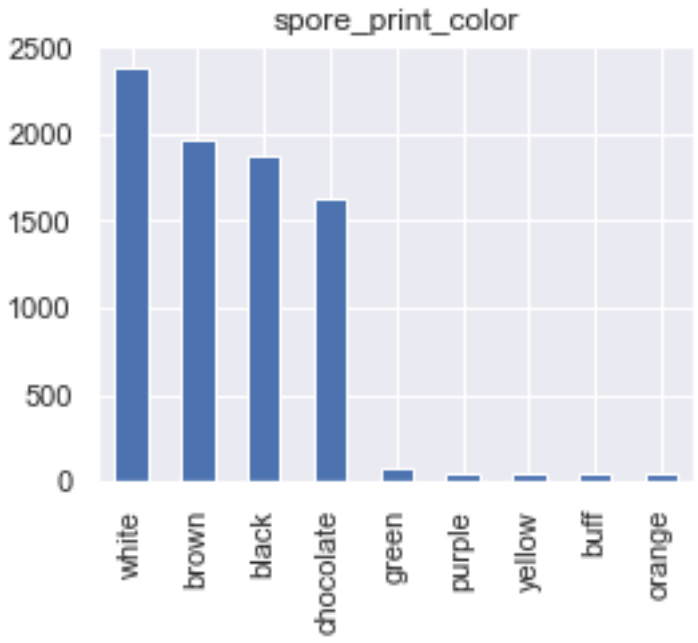

stalk_color_below_ring
4000
3000
2000
1000
0
white
pink
gray
brown
buff
orange
red
cinnamon
yellow

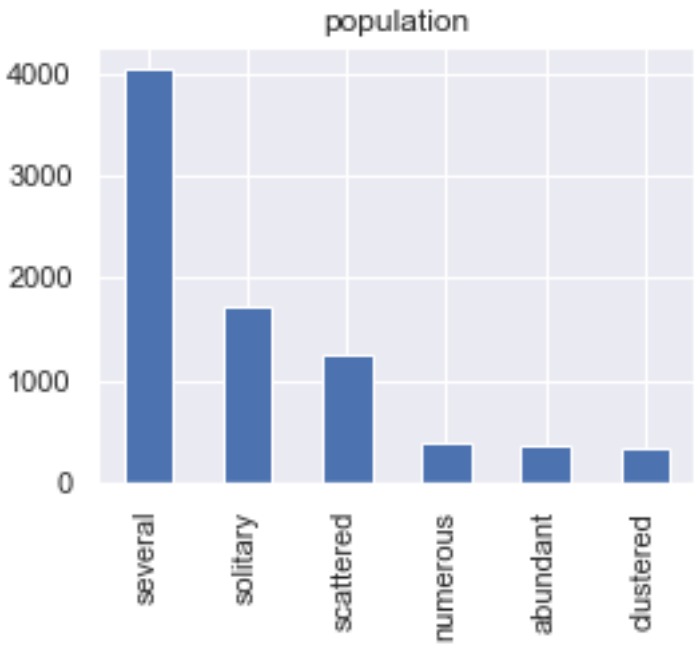

veil_type
8000
6000
4000
2000
0
partial

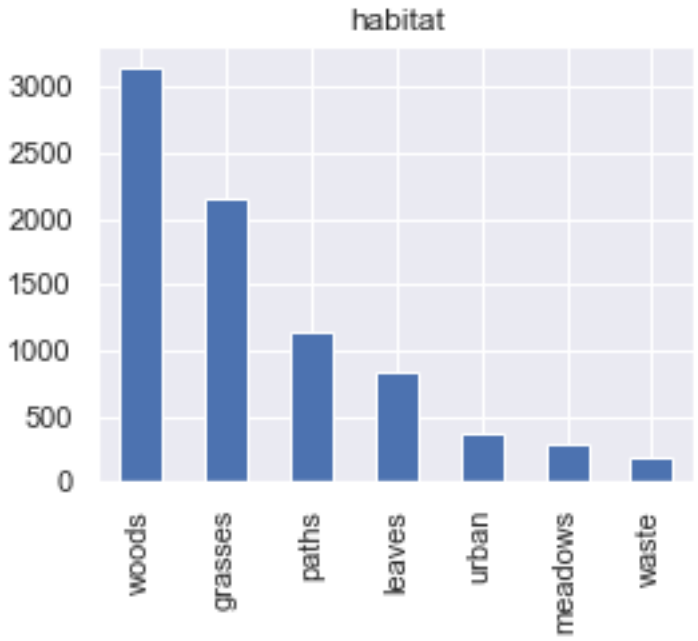

veil_color
8000
6000
4000
2000
0
white
brown
orange
yellow


ring_number
6000
4000
2000
0
one
two
none


ring_type
4000
3000
2000
1000
0
pendant
evanescent
large
flaring
none


spore_print_color
2500
2000
1500
1000
500
0
white
brown
black
chocolate
green
purple
yellow
buff
orange

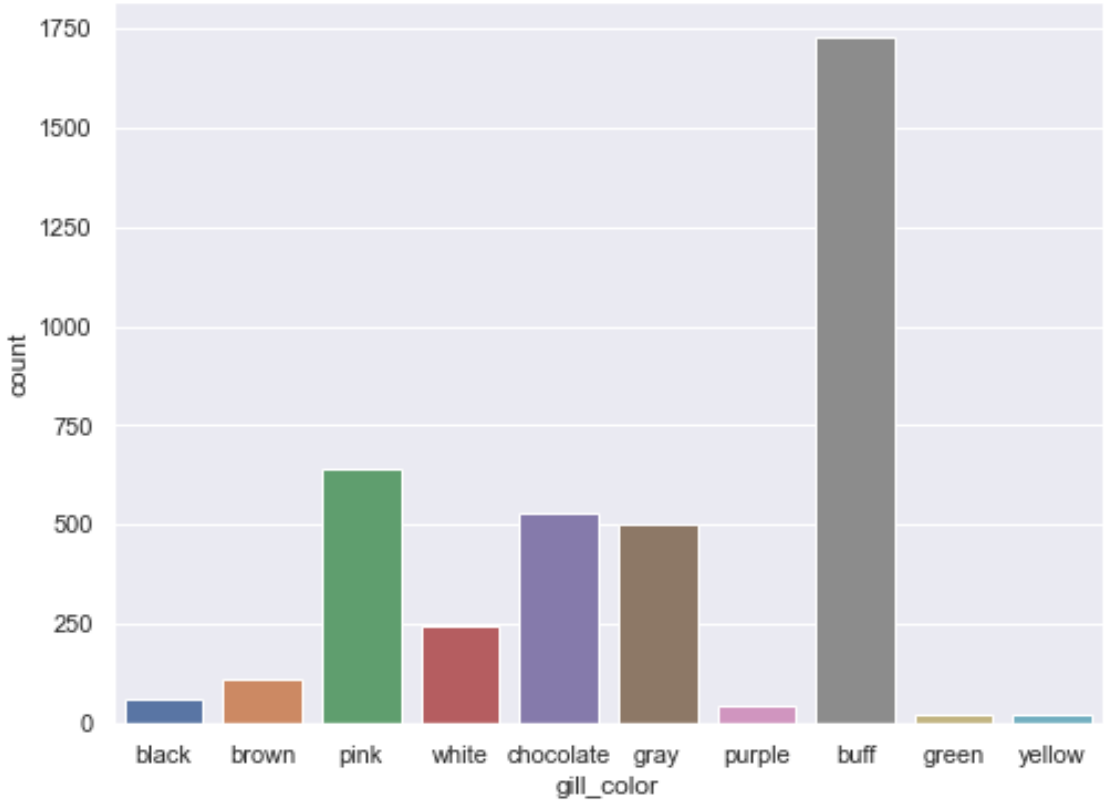




[175]:
```
poison_df = data[data['classes'] == 1]
edible_df = data[data['classes'] == 0]
```

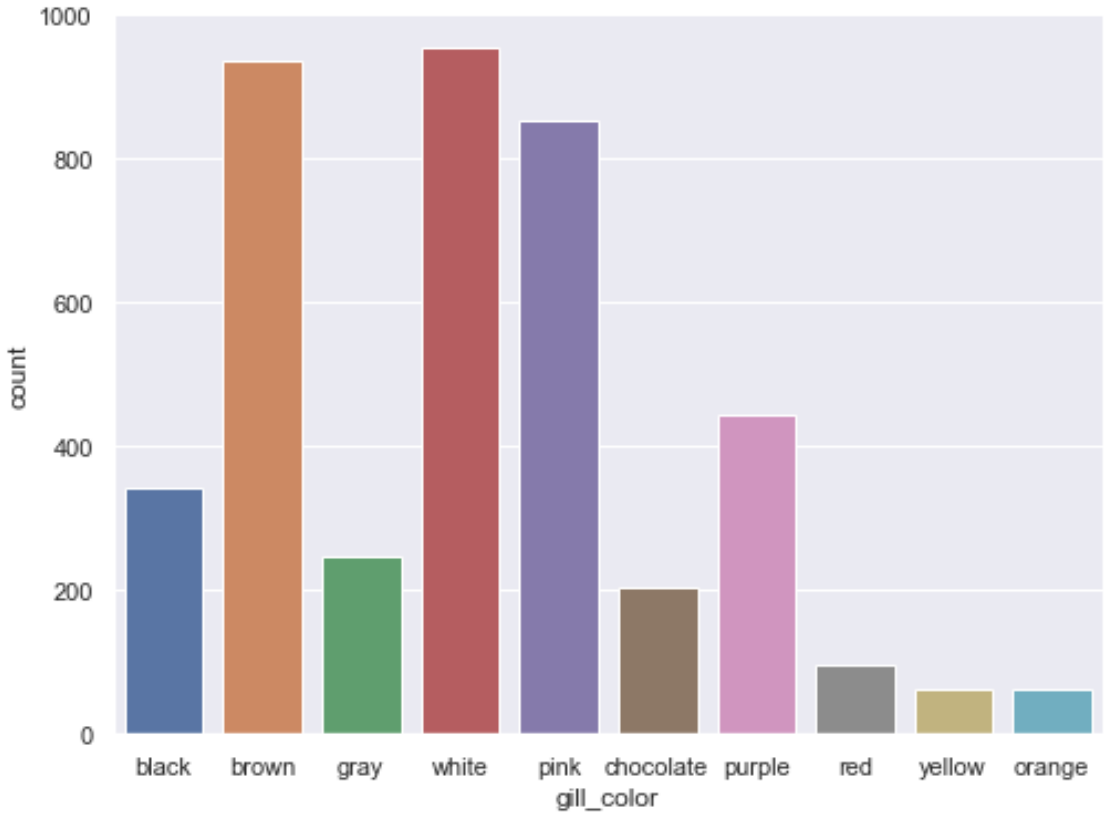
[194]:
```
plt.figure(figsize = (8,6))
sns.countplot(data=poison_df, x='gill_color')
```

[194]: <AxesSubplot:xlabel='gill_color', ylabel='count'>



[195]:
```
plt.figure(figsize = (8,6))
sns.countplot(data=edible_df, x='gill_color')
```

[195]: <AxesSubplot:xlabel='gill_color', ylabel='count'>



[178]:
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=0.33, random_state=25)
```

[179]:
```
print(train.shape)
print(test.shape)
```

```
(5443, 23)
(2681, 23)
```

0.1 Linear model – LOGISTIC REGRESSION

[180]:
```
train_m1.head()
```

[180]:

	classes	cap_shape	cap_color	cap_surface
1423	0	flat	gray	fibrous
6079	1	convex	brown	smooth
8046	0	bell	gray	fibrous
2632	0	convex	gray	scaly
7257	1	knobbed	brown	scaly

[181]:
```
train.columns.to_list()
```

[181]:
```
['classes',
 'cap_shape',
 'cap_surface',
 'cap_color',
 'bruises',
 'odor',
 'gill_attachment',
 'gill_spacing',
 'gill_size',
 'gill_color',
 'stalk_shape',
 'stalk_root',
 'stalk_surface_above_ring',
 'stalk_surface_below_ring',
 'stalk_color_above_ring',
 'stalk_color_below_ring',
 'veil_type',
 'veil_color',
 'ring_number',
 'ring_type',
 'spore_print_color',
 'population',
 'habitat']
```

[182]:
```
train_m1 = train[['classes', 'cap_shape', 'cap_color', 'cap_surface']]
```

[183]:
```
X1 = train_m1.drop(columns = 'classes')
```

[184]:
```
model1 = sm.logit(formula = 'classes ~' + '+'.join(X1), data = train_m1).fit()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.610403
         Iterations: 35

C:\Users\14132\anaconda3\lib\site-packages\statsmodels\base\model.py:566:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

[185]:
```
model1.summary()
```

[185]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:                classes   No. Observations:                 5443
Model:                          Logit   Df Residuals:                     5425
Method:                           MLE   Df Model:                           17
```

```
Date:                Mon, 16 May 2022   Pseudo R-squ.:                  0.1187
Time:                        12:37:50   Log-Likelihood:                -3322.4
converged:                      False   LL-Null:                       -3770.0
Covariance Type:            nonrobust   LLR p-value:                2.332e-179
==============================================================================
==========
                               coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
Intercept                   -3.6272      0.218    -16.620      0.000      -4.055
-3.199
cap_shape[T.conical]        10.6300     37.639      0.282      0.778     -63.140
84.400
cap_shape[T.convex]          2.4244      0.200     12.116      0.000       2.032
2.817
cap_shape[T.flat]            2.5303      0.202     12.546      0.000       2.135
2.926
cap_shape[T.knobbed]         3.5238      0.222     15.868      0.000       3.089
3.959
cap_shape[T.sunken]        -20.7222   3.59e+04     -0.001      1.000   -7.04e+04
7.04e+04
cap_color[T.buff]            1.0573      0.231      4.581      0.000       0.605
1.510
cap_color[T.cinnamon]       -1.0629      0.421     -2.522      0.012      -1.889
-0.237
cap_color[T.gray]            0.4649      0.084      5.531      0.000       0.300
0.630
cap_color[T.green]         -21.6547   1.75e+04     -0.001      0.999   -3.44e+04
3.43e+04
cap_color[T.pink]            1.1347      0.236      4.805      0.000       0.672
1.598
cap_color[T.purple]        -26.1431   1.85e+05     -0.000      1.000   -3.62e+05
3.62e+05
cap_color[T.red]             0.5501      0.087      6.338      0.000       0.380
0.720
cap_color[T.white]          -0.3008      0.107     -2.809      0.005      -0.511
-0.091
cap_color[T.yellow]          1.4265      0.105     13.541      0.000       1.220
1.633
cap_surface[T.grooves]      19.5834   2472.880      0.008      0.994   -4827.172
4866.339
cap_surface[T.scaly]         0.8574      0.074     11.616      0.000       0.713
1.002
cap_surface[T.smooth]        1.1523      0.083     13.843      0.000       0.989
1.315
==============================================================================
```

```
==========
"""
```

[209]:
```
confusion_matrix_data_logit(X1, train_m1.classes, model1, cutoff=0.5)
```

```
Accuracy =  66.89325739481903
FNR =  32.839787395596055
Confusion matrix =
           Predicted 0  Predicted 1
Actual 0       1872.0        937.0
Actual 1        865.0       1769.0
```

[209]:
```
' '
```

0.2 Non-linear model – DECISION TREE

[186]:
```
train_m2 = pd.get_dummies(train)
test_m2 = pd.get_dummies(test)
```

[187]:
```
X2 = train_m2.drop(columns = 'classes')
X2test = test_m2.drop(columns = 'classes')
y2 = train_m2['classes']
y2test = test_m2['classes']
```

[188]:
```
model2 = DecisionTreeClassifier(random_state=1, max_depth=3)
model2.fit(X2, y2)
```

[188]:
```
DecisionTreeClassifier(max_depth=3, random_state=1)
```

[189]:
```
#Function to compute confusion matrix and prediction accuracy on test/train␣
 ↪data -- Decision Tree
def confusion_matrix_data(data,actual_values,model,cutoff=0.5):
#Predict the values using the Logit model
    pred_values = model.predict_proba(data)[:,1]
# Specify the bins
    bins=np.array([0,cutoff,1])
#Confusion matrix
    cm = np.histogram2d(actual_values, pred_values, bins=bins)[0]
    cm_df = pd.DataFrame(cm)
    cm_df.columns = ['Predicted 0','Predicted 1']
    cm_df = cm_df.rename(index={0: 'Actual 0',1:'Actual 1'})
# Calculate the accuracy
    accuracy = 100*(cm[0,0]+cm[1,1])/cm.sum()
    fnr = 100*(cm[1,0])/(cm[1,0]+cm[1,1])
    print("Accuracy = ", accuracy)
    print("FNR = ", fnr)
    print("Confusion matrix = \n", cm_df)
    return (" ")
```

[208]:
```
#Function to compute confusion matrix and prediction accuracy on test/train␣
 ↪data -- Decision Tree
def confusion_matrix_data_logit(data,actual_values,model,cutoff=0.5):
#Predict the values using the Logit model
    pred_values = model.predict(data)
# Specify the bins
    bins=np.array([0,cutoff,1])
#Confusion matrix
    cm = np.histogram2d(actual_values, pred_values, bins=bins)[0]
    cm_df = pd.DataFrame(cm)
    cm_df.columns = ['Predicted 0','Predicted 1']
    cm_df = cm_df.rename(index={0: 'Actual 0',1:'Actual 1'})
# Calculate the accuracy
    accuracy = 100*(cm[0,0]+cm[1,1])/cm.sum()
    fnr = 100*(cm[1,0])/(cm[1,0]+cm[1,1])
    print("Accuracy = ", accuracy)
    print("FNR = ", fnr)
    print("Confusion matrix = \n", cm_df)
    return (" ")
```

[190]:
```
confusion_matrix_data(X2, train_m2.classes, model2, cutoff=0.5)
```

```
Accuracy =  98.4751056402719
FNR =  0.26575550493545935
Confusion matrix =
           Predicted 0  Predicted 1
Actual 0       2733.0         76.0
Actual 1          7.0       2627.0
```

[190]:
```
' '
```