The Data

Mushroom Hunting: Edible or Poisonous?

Data Source: https://archive.ics.uci.edu/ml/datasets/Mushroom (https://archive.ics.uci.edu/ml/datasets/Mushroom)

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like ``leaflets three, let it be'' for Poisonous Oak and Ivy.

Attribute Information:

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s

14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
16. veil-type: partial=p,universal=u
17. veil-color: brown=n,orange=o,white=w,yellow=y
18. ring-number: none=n,one=o,two=t
19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

Imports

In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("mushrooms.csv")
df.head()
```

Out[2]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	st co abo r
0	p	x	s	n	t	p	f	c	n	k	...	s	
1	e	x	s	y	t	a	f	c	b	k	...	s	
2	e	b	s	w	t	l	f	c	b	n	...	s	
3	p	x	y	w	t	p	f	c	n	n	...	s	
4	e	x	s	g	f	n	f	w	b	k	...	s	

5 rows × 23 columns

X & y

In [3]:
```
X = pd.get_dummies(df.drop('class',axis=1),drop_first=True)
y = df['class']
```

Train Test Split

In [4]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_
```

Modelling - Gradient Boosting

In [5]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [6]:
```
help(GradientBoostingClassifier)
```

```
...
```

In [7]:
```
# first model should be with default parameters
gb_model = GradientBoostingClassifier()
```

In [8]:
```
# fit the model with training data set
gb_model.fit(X_train,y_train)
```

Out[8]:
```
GradientBoostingClassifier()
```

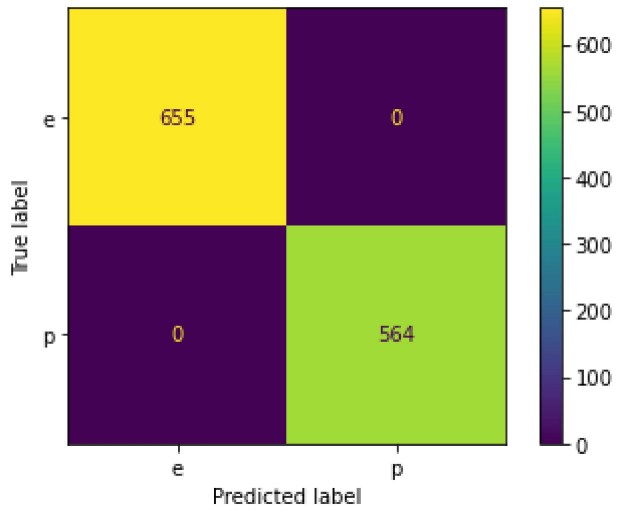
In [9]:
```
#predict over X_train & predict over X_test
y_pred_train = gb_model.predict(X_train)
y_pred_test = gb_model.predict(X_test)
```

In [10]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_train,y_pred_train)) # train accuracy
print(accuracy_score(y_test,y_pred_test))   # test accuracy
```

```
1.0
1.0
```

In [11]:
```
from sklearn.metrics import plot_confusion_matrix
print(plot_confusion_matrix(gb_model,X_test,y_test))
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x0000
02653374A610>
```



In [12]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           e       1.00      1.00      1.00       655
           p       1.00      1.00      1.00       564

    accuracy                           1.00      1219
   macro avg       1.00      1.00      1.00      1219
weighted avg       1.00      1.00      1.00      1219
```

In [13]:
```
from sklearn.model_selection import GridSearchCV
```

In [14]:
```
gb_model = GradientBoostingClassifier()
param_grid = {"n_estimators":[1,5,10,20,40,100],'max_depth':[3,4,5,6]}
```

In [15]:
```
grid = GridSearchCV(gb_model,param_grid,cv=5,scoring='accuracy')
```

In [16]:
```
grid.fit(X_train,y_train) ### Fit to Training Data with CV Search
```

Out[16]:
```
GridSearchCV(cv=5, estimator=GradientBoostingClassifier(),
             param_grid={'max_depth': [3, 4, 5, 6],
                         'n_estimators': [1, 5, 10, 20, 40, 100]},
             scoring='accuracy')
```

In [17]:
```
grid.best_params_
```

Out[17]:
```
{'max_depth': 3, 'n_estimators': 100}
```

In [18]:
```
predictions = grid.predict(X_test)
```

In [19]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           e       1.00      1.00      1.00       655
           p       1.00      1.00      1.00       564

    accuracy                           1.00      1219
   macro avg       1.00      1.00      1.00      1219
weighted avg       1.00      1.00      1.00      1219
```

Feature Importance

In [20]:
```
grid.best_estimator_.feature_importances_
```

Out[20]:
```
array([2.91150176e-04, 4.16840144e-17, 2.62963075e-21, 0.00000000e+00,
       8.95272815e-17, 1.04524302e-03, 1.18312069e-05, 5.06011038e-06,
       7.30497335e-18, 2.25556577e-18, 4.15359568e-18, 0.00000000e+00,
       9.62031098e-17, 4.29662881e-21, 1.12698461e-21, 7.74545549e-05,
       2.31056241e-03, 5.15112966e-02, 6.24175887e-04, 1.04679173e-02,
       1.82499853e-02, 3.14895349e-03, 6.14744334e-01, 8.63939501e-03,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.25092906e-02,
       1.13044167e-02, 0.00000000e+00, 2.11998718e-18, 6.27556095e-18,
       1.93736167e-17, 0.00000000e+00, 0.00000000e+00, 3.24319756e-18,
       5.31466406e-16, 0.00000000e+00, 1.74072634e-17, 4.93290554e-18,
       3.98031006e-18, 1.57722440e-04, 1.36013077e-01, 0.00000000e+00,
       2.47258825e-02, 6.21308869e-04, 7.82751635e-06, 4.95063766e-06,
       3.60117275e-05, 6.09765605e-05, 4.25423158e-02, 9.96508051e-05,
       0.00000000e+00, 0.00000000e+00, 8.96840865e-04, 0.00000000e+00,
       0.00000000e+00, 5.03575357e-03, 0.00000000e+00, 2.09684458e-03,
       0.00000000e+00, 0.00000000e+00, 5.33104127e-05, 0.00000000e+00,
       0.00000000e+00, 5.62862863e-04, 3.02342639e-03, 0.00000000e+00,
       0.00000000e+00, 1.10295205e-04, 4.73681003e-05, 6.76478844e-04,
       8.43895349e-05, 0.00000000e+00, 2.69530010e-07, 5.55122482e-04,
       1.09004788e-02, 2.13873553e-04, 2.09308309e-04, 0.00000000e+00,
       3.04953583e-02, 4.10000880e-03, 4.86768755e-04, 0.00000000e+00,
       1.21311432e-03, 0.00000000e+00, 7.26245568e-08, 1.34754549e-05,
       3.74013965e-06, 4.54948477e-16, 0.00000000e+00, 7.75007592e-17,
       0.00000000e+00, 1.00485103e-05, 0.00000000e+00])
```

In [21]:
```
imp_feats = pd.DataFrame(index=X.columns,data=grid.best_estimator_.feature_import
imp_feats
```

Out[21]:

	Importance
cap-shape_c	2.911502e-04
cap-shape_f	4.168401e-17
cap-shape_k	2.629631e-21
cap-shape_s	0.000000e+00
cap-shape_x	8.952728e-17
...	...
habitat_l	0.000000e+00
habitat_m	7.750076e-17
habitat_p	0.000000e+00
habitat_u	1.004851e-05
habitat_w	0.000000e+00

95 rows × 1 columns

In [22]:
```
imp_feats.sort_values("Importance",ascending=False)
```

Out[22]:

	Importance
odor_n	0.614744
stalk-root_c	0.136013
bruises_t	0.051511
stalk-surface-below-ring_y	0.042542
spore-print-color_r	0.030495
...	...
veil-color_o	0.000000
veil-color_w	0.000000
gill-color_e	0.000000
odor_y	0.000000
habitat_w	0.000000

95 rows × 1 columns

In [23]:
```
imp_feats.describe()
```

Out[23]:

	Importance
count	9.500000e+01
mean	1.052632e-02
std	6.465008e-02
min	0.000000e+00
25%	0.000000e+00
50%	7.262456e-08
75%	5.920859e-04
max	6.147443e-01

In [24]:
```
imp_feats = imp_feats[imp_feats['Importance'] > 0.000527]
```

In [25]:
```
plt.figure(figsize=(14,6),dpi=200)
sns.barplot(data=imp_feats.sort_values('Importance'),x=imp_feats.index,y='Importa
plt.xticks(rotation=90);
```