Business Problem

Goal here is to see if we can harness the power of machine learning and boosting to help create not just a predictive model, but a general guideline for features people should look out for when picking mushrooms.

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	a
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

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

In [3]:
```
df.info()
```

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

EDA

In [ ]:
```

```

In [ ]:
```

```

X & y

In [4]:
```
X = pd.get_dummies(df.drop('class',axis=1),drop_first=True)
y = df['class']
```

In [5]:
```
X.shape
```

Out[5]:
```
(8124, 95)
```

Train Test Split

In [6]:
```
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, rand
om_state=101)
```

Modelling

Gradient Boosting with default parameters

In [7]:
```
from sklearn.ensemble import GradientBoostingClassifier

gb_model = GradientBoostingClassifier()

gb_model.fit(X_train,y_train)
```

Out[7]:
```
GradientBoostingClassifier()
```

prediction

In [8]:
```
y_pred_train = gb_model.predict(X_train)

y_pred_test = gb_model.predict(X_test)
```

Evaluation

Accuracy

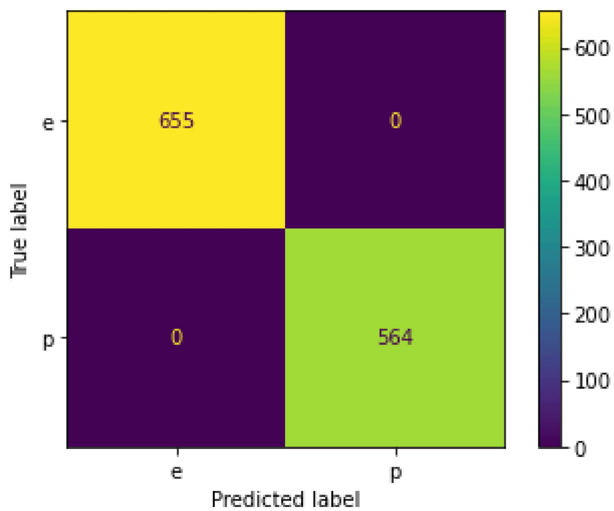
In [9]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_train,y_pred_train)) # train accuracy
print(accuracy_score(y_test,y_pred_test))   # test accuracy
```

```
1.0
1.0
```

Confusion Matrix

In [10]:
```
from sklearn.metrics import plot_confusion_matrix
print(plot_confusion_matrix(gb_model,X_test,y_test))
plt.show()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x00
0001F2CD752820>
```



Classification Report

In [11]:
```
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
```

Cross Validation Score

In [12]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(gb_model,X,y,cv=5)
print("Cross Validation Score:",scores.mean())
```

```
Cross Validation Score: 0.9192312239484653
```

Hyperparameter Tuning

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

estimator= GradientBoostingClassifier()

param_grid = {"n_estimators":[1,5,10,20,40,100],"learning_rate":[0.1,0.2,0.3,
0.5,0.8,1]}

grid = GridSearchCV(estimator,  param_grid, cv=5,  scoring='accuracy')

grid.fit(X_train,y_train)

grid.best_params_
```

Out[13]:
```
{'learning_rate': 0.1, 'n_estimators': 100}
```

Final Model

In [14]:
```
final_model = GradientBoostingClassifier(n_estimators=100,learning_rate=0.1)
final_model.fit(X_train,y_train)

preds_train = final_model.predict(X_train)
preds_test = final_model.predict(X_test)

print("Train Accuracy Score: ", accuracy_score(y_train,preds_train))
print("Test Accuracy Score: ",accuracy_score(y_test,preds_test))
```

```
Train Accuracy Score:  1.0
Test Accuracy Score:  1.0
```

Feature Importance

In [15]:
```
final_model.feature_importances_
```

Out[15]:
```
array([2.91150176e-04, 4.17245225e-16, 0.00000000e+00, 0.00000000e+00,
       1.22159661e-16, 1.04652037e-03, 1.17118353e-17, 3.78276239e-06,
       3.48346246e-18, 1.53829642e-17, 1.13678937e-17, 0.00000000e+00,
       3.73421008e-17, 1.81961056e-21, 0.00000000e+00, 5.60405971e-07,
       2.31055039e-03, 5.30392238e-02, 1.84253604e-04, 1.00808510e-02,
       1.82499853e-02, 3.66222069e-04, 6.14762854e-01, 9.20844491e-04,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 1.30047904e-02,
       1.03950811e-02, 0.00000000e+00, 5.22781393e-17, 0.00000000e+00,
       4.80274821e-17, 0.00000000e+00, 0.00000000e+00, 9.59297515e-18,
       1.16274113e-16, 0.00000000e+00, 1.86446690e-17, 1.71395576e-21,
       2.48611630e-18, 4.58875925e-04, 1.35972688e-01, 7.71855052e-03,
       5.17747365e-02, 4.65375385e-04, 6.12113083e-06, 1.15245842e-04,
       0.00000000e+00, 0.00000000e+00, 1.75588775e-02, 2.97761690e-06,
       0.00000000e+00, 0.00000000e+00, 1.22682934e-03, 0.00000000e+00,
       0.00000000e+00, 6.23301226e-04, 7.74443653e-05, 1.26707456e-03,
       0.00000000e+00, 0.00000000e+00, 5.33104127e-05, 0.00000000e+00,
       0.00000000e+00, 1.42943863e-03, 3.02342639e-03, 0.00000000e+00,
       1.35380870e-07, 0.00000000e+00, 2.58827766e-03, 0.00000000e+00,
       6.46948291e-05, 1.76797782e-05, 1.05423536e-05, 5.21100751e-04,
       1.13072397e-02, 2.14184641e-04, 2.08997222e-04, 0.00000000e+00,
       3.04953583e-02, 4.10000880e-03, 2.37566832e-03, 0.00000000e+00,
       1.17406172e-03, 0.00000000e+00, 6.07840776e-08, 1.35041668e-05,
       4.67493807e-04, 1.55987642e-17, 0.00000000e+00, 5.45257692e-17,
       0.00000000e+00, 1.00485103e-05, 0.00000000e+00])
```

In [16]:
```
imp_feats = pd.DataFrame(index=X.columns,data=grid.best_estimator_.feature_importances_,columns=['Importance'])
imp_feats
```

Out[16]:

	Importance
cap-shape_c	2.911502e-04
cap-shape_f	1.772087e-16
cap-shape_k	0.000000e+00
cap-shape_s	0.000000e+00
cap-shape_x	1.021728e-16
...	...
habitat_l	0.000000e+00
habitat_m	2.673433e-17
habitat_p	0.000000e+00
habitat_u	1.004851e-05
habitat_w	0.000000e+00

95 rows × 1 columns

In [17]:
```
imp_feats = imp_feats[imp_feats['Importance'] > 0.01]
```

In [18]:
```
plt.figure(figsize=(14,6),dpi=200)
sns.barplot(data=imp_feats.sort_values('Importance'),x=imp_feats.index,y='Importance')
plt.xticks(rotation=90);
```