In [1]:
```
# Installing XGBoost using Anaconda prompt
# "conda install -c anaconda py-xgboost"
```

In [2]:
```
# Installing XGBoost using jupyter notebook
# !pip install xgboost
```

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 [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.simplefilter("ignore")
```

In [4]:
```
df = pd.read_csv("mushrooms.csv")
df.head()
```

Out[4]:

	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 [5]:
```
X = pd.get_dummies(df.drop('class',axis=1),drop_first=True)
y = df['class']
```

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, random_
```

Modelling - Gradient Boosting

In [7]:
```
from xgboost import XGBClassifier
```

In [8]:
```
help(XGBClassifier)
```

...

In [9]:
```
# first model should be with default parameters
xgb_model = XGBClassifier()
```

In [10]:
```
# fit the model with training data set
xgb_model.fit(X_train,y_train)
```

```
[14:18:57] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[10]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [11]:
```
#predict over X_train & predict over X_test
y_pred_train = xgb_model.predict(X_train)
y_pred_test = xgb_model.predict(X_test)
```

In [12]:
```
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
```

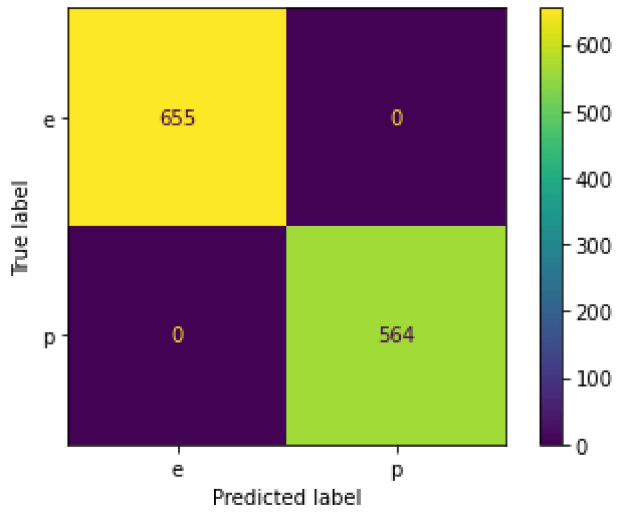
In [13]:
```
# Applying k-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = xgb_model, X = X_train, y = y_train, cv
accuracies.mean()
```

```
[14:18:58] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
[14:18:59] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
[14:19:00] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
[14:19:01] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
[14:19:01] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.
0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[13]:
```
1.0
```

In [14]:
```
from sklearn.metrics import plot_confusion_matrix
print(plot_confusion_matrix(xgb_model,X_test,y_test))
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x0000
01B3B5548BB0>
```



In [15]:
```
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 [16]:
```
from sklearn.model_selection import GridSearchCV
```

In [17]:
```
xgb_model = XGBClassifier()
param_grid = {"n_estimators":[1,5,10,20,40,100],'max_depth':[3,4,5,6],'gamma':[0,
```

In [19]:
```
grid = GridSearchCV(xgb_model,param_grid,cv=5,scoring='accuracy')
grid.fit(X_train,y_train)
```

...

In [21]:
```
grid.best_params_
```

Out[21]:
```
{'gamma': 0, 'max_depth': 3, 'n_estimators': 40}
```

In [22]:
```
predictions = grid.predict(X_test)
```

In [23]:
```
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 [24]:
```
grid.best_estimator_.feature_importances_
```

...

In [25]:
```
imp_feats = pd.DataFrame(index=X.columns,data=grid.best_estimator_.feature_import
imp_feats
```

Out[25]:

	Importance
cap-shape_c	0.000000
cap-shape_f	0.000000
cap-shape_k	0.000000
cap-shape_s	0.000000
cap-shape_x	0.000055
...	...
habitat_l	0.000000
habitat_m	0.000000
habitat_p	0.000000
habitat_u	0.000000
habitat_w	0.000000

95 rows × 1 columns

In [26]:
```
imp_feats.sort_values("Importance",ascending=False)
```

Out[26]:

	Importance
stalk-root_c	0.282541
odor_n	0.254216
stalk-root_r	0.175081
odor_l	0.092742
bruises_t	0.040733
...	...
gill-color_r	0.000000
gill-color_p	0.000000
gill-color_o	0.000000
gill-color_n	0.000000
habitat_w	0.000000

95 rows × 1 columns

In [27]:
```
imp_feats.describe()
```

Out[27]:

	Importance
count	95.000000
mean	0.010526
std	0.043377
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000989
max	0.282541

In [28]:
```
imp_feats = imp_feats[imp_feats['Importance'] > 0.000527]
```

In [29]:
```
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);
```