AdaBoost

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

Goal

THIS IS IMPORTANT, THIS IS NOT OUR TYPICAL PREDICTIVE MODEL!

Our general 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.

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

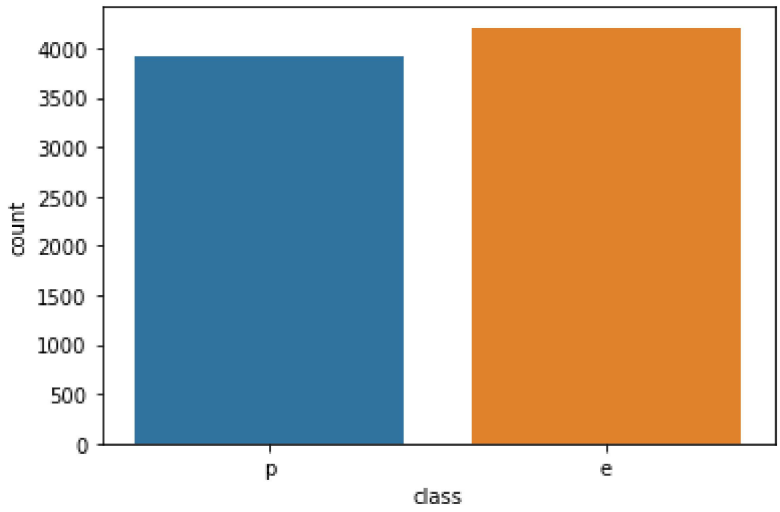
In [3]:
```
df.info()
```

```
<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 [4]:
```
sns.countplot(data=df,x='class')
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x258d57c9df0>
```



In [5]:
```
df.describe()
```

Out[5]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk surface below ring
count	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	...	812
unique	2	6	4	10	2	9	2	2	2	12	...	
top	e	x	y	n	f	n	f	c	b	b	...	
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	...	493

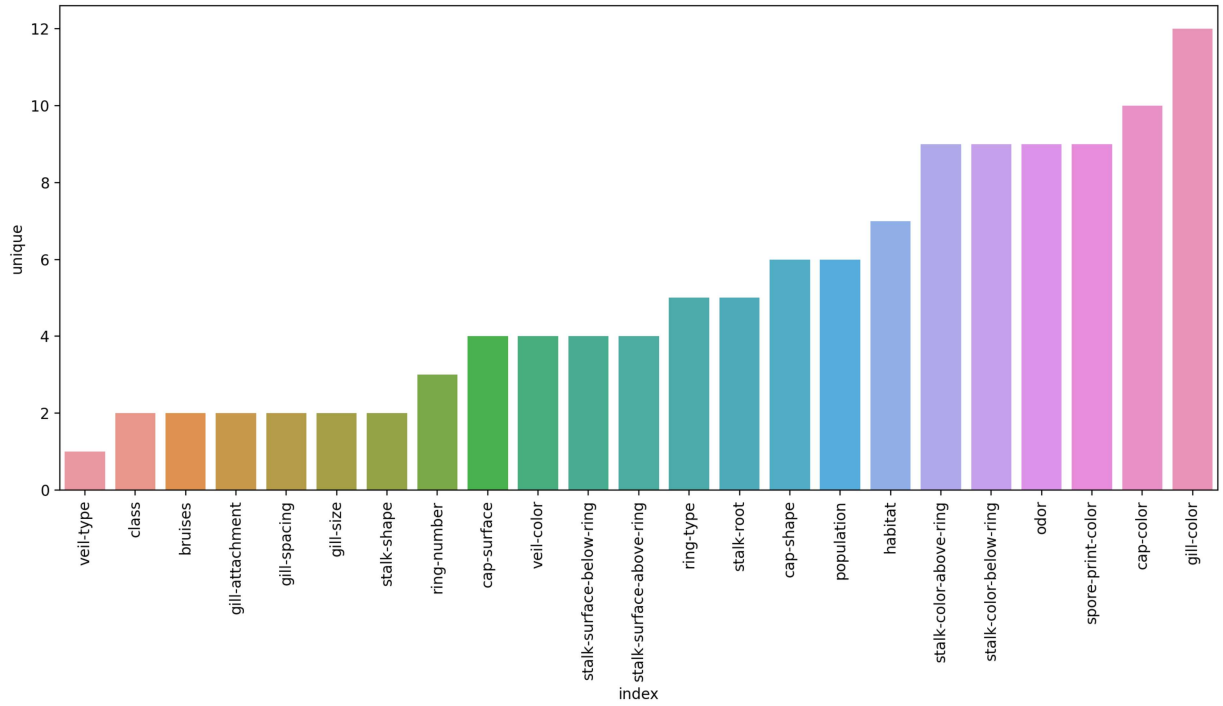
4 rows × 23 columns

In [6]:
```
df.describe().transpose()
```

Out[6]:

	count	unique	top	freq
class	8124	2	e	4208
cap-shape	8124	6	x	3656
cap-surface	8124	4	y	3244
cap-color	8124	10	n	2284
bruises	8124	2	f	4748
odor	8124	9	n	3528
gill-attachment	8124	2	f	7914
gill-spacing	8124	2	c	6812
gill-size	8124	2	b	5612
gill-color	8124	12	b	1728
stalk-shape	8124	2	t	4608
stalk-root	8124	5	b	3776
stalk-surface-above-ring	8124	4	s	5176
stalk-surface-below-ring	8124	4	s	4936
stalk-color-above-ring	8124	9	w	4464
stalk-color-below-ring	8124	9	w	4384
veil-type	8124	1	p	8124
veil-color	8124	4	w	7924
ring-number	8124	3	o	7488
ring-type	8124	5	p	3968
spore-print-color	8124	9	w	2388
population	8124	6	v	4040
habitat	8124	7	d	3148

In [7]:
```
plt.figure(figsize=(14,6),dpi=200)
sns.barplot(data=df.describe().transpose().reset_index().sort_values('unique'),x=
plt.xticks(rotation=90);
```



Train Test Split

In [8]:
```
#X = df.drop('class',axis=1)
```

In [9]:
```
#X = pd.get_dummies(X,drop_first=True)
```

In [10]:
```
X = pd.get_dummies(df.drop('class',axis=1),drop_first=True)
y = df['class']
```

In [11]:
```
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_
```

Modeling

In [12]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [13]:
```
model = AdaBoostClassifier(n_estimators=1)
```

In [14]:
```
model.fit(X_train,y_train)
```

Out[14]:
```
AdaBoostClassifier(n_estimators=1)
```

Evaluation

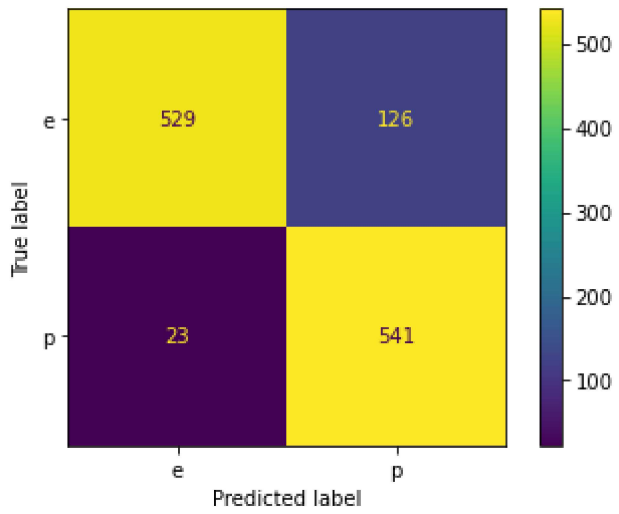
In [15]:
```
from sklearn.metrics import classification_report,plot_confusion_matrix,accuracy_
```

In [16]:
```
predictions = model.predict(X_test)
```

In [18]:
```
accuracy_score(y_test,predictions)
```

Out[18]:
```
0.8777686628383922
```

In [19]:
```
plot_confusion_matrix(model,X_test,y_test)
plt.show()
```



In [20]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           e       0.96      0.81      0.88       655
           p       0.81      0.96      0.88       564

    accuracy                           0.88      1219
   macro avg       0.88      0.88      0.88      1219
weighted avg       0.89      0.88      0.88      1219
```

In [21]:
```
model.feature_importances_
```

Out[21]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Analyzing performance as more weak learners are added.

In [22]:
```
error_rates = []

for n in range(1,96):

    model = AdaBoostClassifier(n_estimators=n)
    model.fit(X_train,y_train)
    preds = model.predict(X_test)
    err = 1 - accuracy_score(y_test,preds)

    error_rates.append(err)
```

In [23]:
```
plt.figure(figsize=(40,15))
plt.plot(range(1,96),error_rates)
plt.xticks(list(range(1,96)))
plt.show()
```

Final Model

In [24]:
```
final_model = AdaBoostClassifier(n_estimators=17)
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
```

In [25]:
```
final_model.feature_importances_
```

Out[25]:
```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.05882353, 0.05882353, 0.05882353,
       0.        , 0.        , 0.11764706, 0.05882353, 0.        ,
       0.        , 0.        , 0.        , 0.11764706, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.05882353, 0.        , 0.        , 0.        , 0.        ,
       0.05882353, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.05882353, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.05882353, 0.        , 0.11764706, 0.        , 0.05882353,
       0.        , 0.        , 0.11764706, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ])
```

In [26]:
```
feats = pd.DataFrame(index=X.columns,data=final_model.feature_importances_,columr
feats
```

Out[26]:

	Importance
cap-shape_c	0.0
cap-shape_f	0.0
cap-shape_k	0.0
cap-shape_s	0.0
cap-shape_x	0.0
...	...
habitat_l	0.0
habitat_m	0.0
habitat_p	0.0
habitat_u	0.0
habitat_w	0.0

95 rows × 1 columns

In [27]:
```
imp_feats = feats[feats['Importance']>0]
```

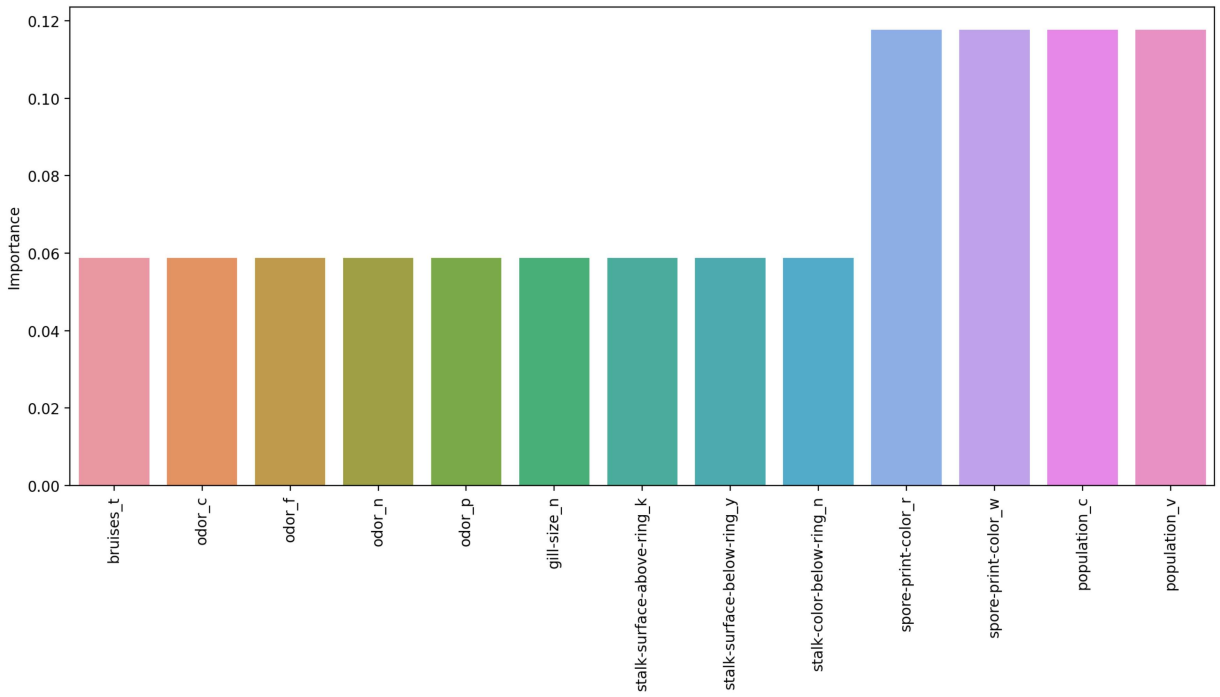
In [28]:
```
imp_feats.sort_values("Importance")
```

Out[28]:

	Importance
bruises_t	0.058824
odor_c	0.058824
odor_f	0.058824
odor_p	0.058824
stalk-surface-above-ring_k	0.058824
stalk-surface-below-ring_y	0.058824
stalk-color-below-ring_n	0.058824
spore-print-color_r	0.058824
population_c	0.058824
odor_n	0.117647
gill-size_n	0.117647
spore-print-color_w	0.117647
population_v	0.117647

In [30]:
```
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)
plt.show()
```



Interesting to see how the importance of the features shift as more are allowed to be added in! But remember these are all weak learner stumps, and feature importance is available for all the tree methods!