

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	odor
0	p	x	s	n	t	p	f	c	n	k	...	s	a
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

Data

Mushroom : Edible or Poisonous?

Data Source: <https://archive.ics.uci.edu/ml/datasets/Mushroom>
[\(https://archive.ics.uci.edu/ml/datasets/Mushroom\)](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:

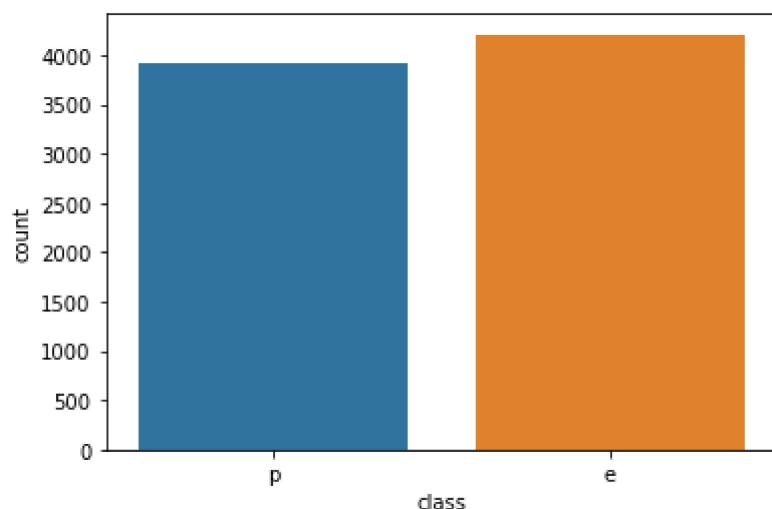
1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
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=
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 [4]: sns.countplot(data=df,x='class')
plt.show()
```



In [5]: df.describe()

Out[5]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	st bel
count	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8124	8
unique	2	6	4	10	2	9	2	2	2	2	12	...
top	e	x	y	n	f	n	f	c	b	b	b	...
freq	4208	3656	3244	2284	4748	3528	7914	6812	5612	1728	...	4

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

X & y

```
In [7]: X = pd.get_dummies(df.drop('class',axis=1),drop_first=True)
y = df['class']
```

```
In [8]: X
```

Out[8]:

	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_g	cap-surface_s	cap-surface_y	cap-color_c
0	0	0	0	0	1	0	1	0	0
1	0	0	0	0	1	0	1	0	0
2	0	0	0	0	0	0	1	0	0
3	0	0	0	0	1	0	0	1	0
4	0	0	0	0	1	0	1	0	0
...
8119	0	0	1	0	0	0	1	0	0
8120	0	0	0	0	1	0	1	0	0
8121	0	1	0	0	0	0	1	0	0
8122	0	0	1	0	0	0	0	1	0
8123	0	0	0	0	1	0	1	0	0

8124 rows × 95 columns

```
In [9]: X.shape,y.shape
```

Out[9]: ((8124, 95), (8124,))

Train/Test Split

```
In [10]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
```

Modeling

AdaBoost Classifier with default parameters

```
In [11]: from sklearn.ensemble import AdaBoostClassifier
```

```
In [12]: model = AdaBoostClassifier()
```

```
In [13]: model.fit(X_train,y_train)
```

```
Out[13]: AdaBoostClassifier()
```

Prediction

```
In [14]: ypred_train=model.predict(X_train)
```

```
In [15]: predictions = model.predict(X_test)
```

Evaluation

Accuracy

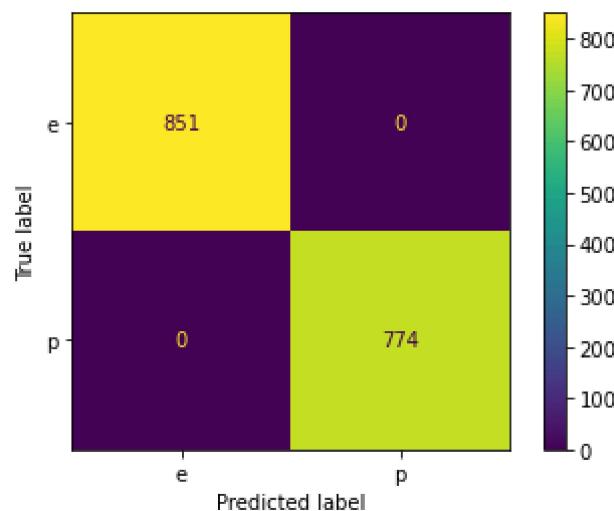
```
In [16]: from sklearn.metrics import accuracy_score  
print("Train accuracy:",accuracy_score(y_train,ypred_train))  
print("Test accuracy:",accuracy_score(y_test,predictions))
```

Train accuracy: 1.0

Test accuracy: 1.0

Confusion Matrix

```
In [17]: from sklearn.metrics import plot_confusion_matrix  
plot_confusion_matrix(model,X_test,y_test)  
plt.show()
```



Classification Report

```
In [18]: from sklearn.metrics import classification_report  
print(classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
e	1.00	1.00	1.00	851
p	1.00	1.00	1.00	774
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
weighted avg	1.00	1.00	1.00	1625

Cross validation Score

```
In [19]: from sklearn.model_selection import cross_val_score  
scores = cross_val_score(model,X,y,cv=5)  
print("Cross Validation Score:",scores.mean())
```

Cross Validation Score: 0.9192312239484653

Hyperparameter Tuning

```
In [20]: from sklearn.model_selection import GridSearchCV  
  
estimator = AdaBoostClassifier()  
  
param_grid = {"n_estimators":list(range(1,101))}  
  
grid = GridSearchCV(estimator, param_grid, cv=5, scoring='accuracy')  
  
grid.fit(X_train,y_train)  
  
grid.best_params_
```

Out[20]: {'n_estimators': 20}

Final Model

```
In [21]: final_model = AdaBoostClassifier(n_estimators=20)
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 [22]: final_model.feature_importances_
```

```
Out[22]: array([0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ,
 0. , 0. , 0. , 0. , 0.05, 0. , 0.05, 0.05, 0.05, 0. , 0. ,
 0.1 , 0.05, 0. , 0. , 0. , 0.05, 0.15, 0. , 0. , 0. , 0. ,
 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ,
 0. , 0.05, 0. , 0. , 0. , 0. , 0.05, 0. , 0. , 0. , 0. ,
 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.05, 0. , 0. , 0. ,
 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ,
 0. , 0. , 0. , 0.05, 0. , 0.1 , 0. , 0. , 0.05, 0. , 0. ,
 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.1 ,
```

```
In [23]: feats = pd.DataFrame(index=X.columns,data=final_model.feature_importances_,columns=[ 'Importance'])
feats
```

Out[23]:

	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 [24]: imp_feats = feats[feats['Importance']>0]
imp_feats.sort_values("Importance")
```

Out[24]:

	Importance
cap-color_w	0.05
bruises_t	0.05
odor_c	0.05
odor_f	0.05
odor_p	0.05
gill-spacing_w	0.05
stalk-surface-above-ring_k	0.05
stalk-surface-below-ring_y	0.05
stalk-color-below-ring_n	0.05
spore-print-color_r	0.05
population_c	0.05
odor_n	0.10
spore-print-color_w	0.10
population_v	0.10
gill-size_n	0.15

```
In [25]: 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)
plt.show()
```

