AdaBoost

May 28, 2023

1 AdaBoost

1.1 The Data

1.1.1 Mushroom Hunting: Edible or Poisonous?

Data Source: 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

2 Goal / Objetivo

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.

ISSO É IMPORTANTE, NÃO É NOSSO MODELO PREDITIVO TÍPICO!

Nosso objetivo geral aqui é ver se podemos aproveitar o poder do aprendizado de máquina e aumentar para ajudar a criar não apenas um modelo preditivo, mas uma diretriz geral para os recursos que as pessoas devem observar ao colher cogumelos.

2.1 Imports

2.2 Importações

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

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

Load the mushroom dataset from a CSV file

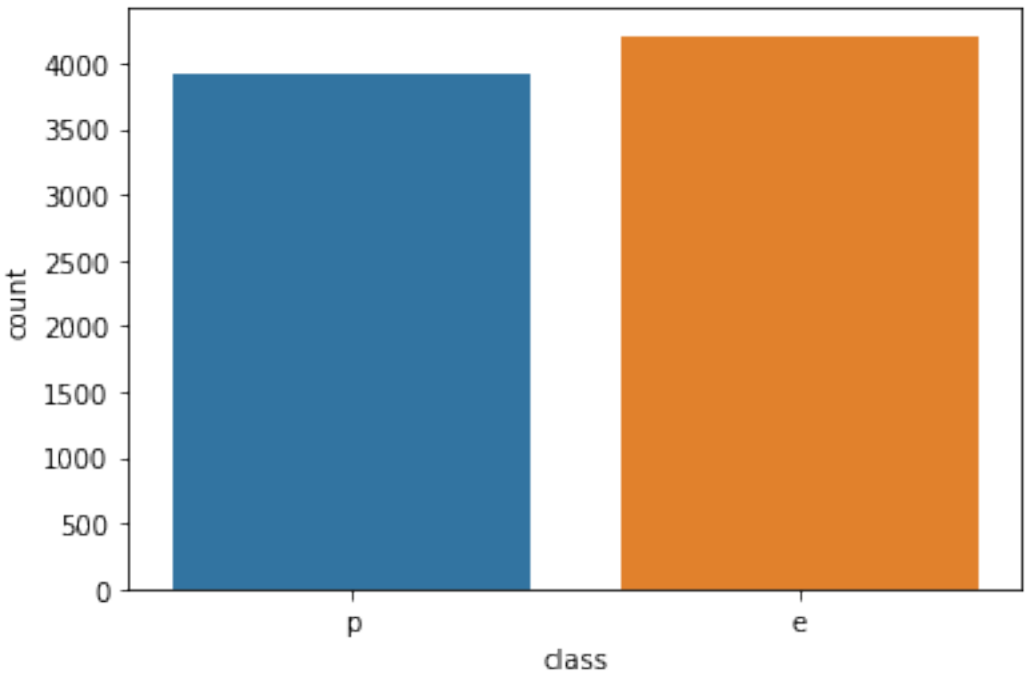
Carrega o conjunto de dados de cogumelos a partir de um arquivo CSV

[2]:
```
df = pd.read_csv("../DATA/mushrooms.csv")
```

3 Exploratory Data Analysis

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

[4]:
```
<AxesSubplot:xlabel='class', ylabel='count'>
```



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

[5]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	…
count	8124	8124	8124	8124	8124	8124	8124	…
unique	2	6	4	10	2	9	2	…
top	e	x	y	n	f	n	f	…
freq	4208	3656	3244	2284	4748	3528	7914	…

	gill-spacing	gill-size	gill-color	…	stalk-surface-below-ring	…
count	8124	8124	8124	…	8124	…
unique	2	2	12	…	4	…
top	c	b	b	…	s	…
freq	6812	5612	1728	…	4936	…

	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	…
count	8124	8124	8124	8124	…
unique	9	9	1	4	…
top	w	w	p	w	…
freq	4464	4384	8124	7924	…

	ring-number	ring-type	spore-print-color	population	habitat
count	8124	8124	8124	8124	8124
unique	3	5	9	6	7
top	o	p	w	v	d

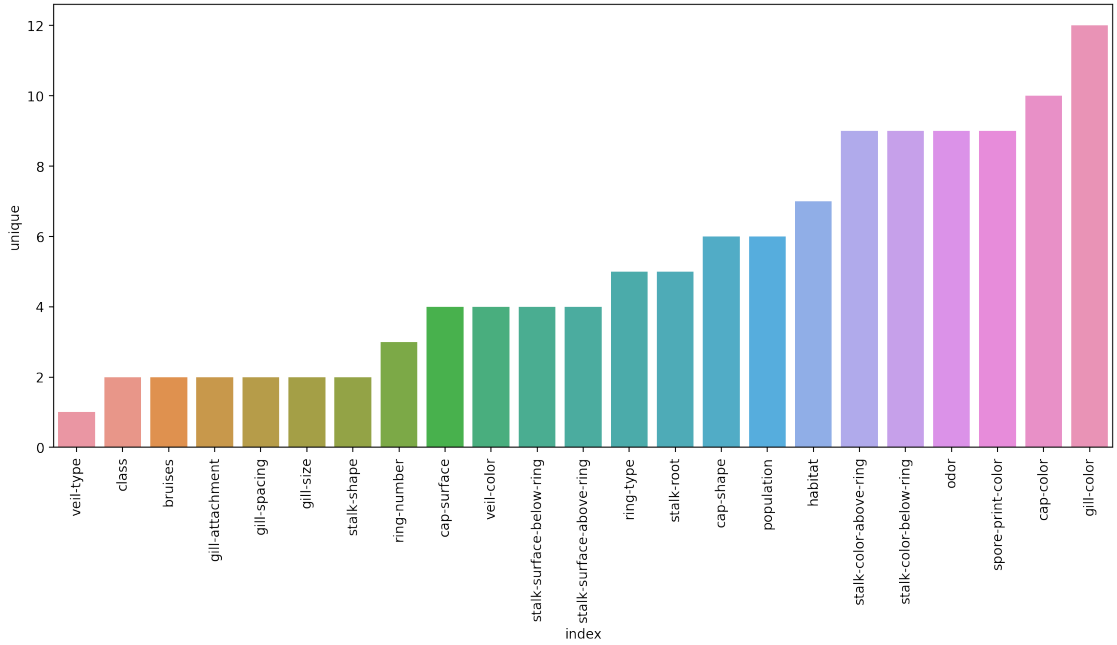
```
freq          7488      3968                2388       4040    3148

[4 rows x 23 columns]
```

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

[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

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



4 Train Test Split

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

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

[10]:
```
y = df['class']
```

[11]:
```
from sklearn.model_selection import train_test_split
```

[12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,␣
↪random_state=101)
```

5 Modeling

[13]:
```
from sklearn.ensemble import AdaBoostClassifier
```

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

[15]:
```
model.fit(X_train,y_train)
```

[15]:
```
AdaBoostClassifier(n_estimators=1)
```

5.1 Evaluation

[16]:
```
from sklearn.metrics import␣
↪classification_report,plot_confusion_matrix,accuracy_score
```

[17]:
```
predictions = model.predict(X_test)
```

[18]:
```
predictions
```

[18]:
```
array(['p', 'e', 'p', …, 'p', 'p', 'e'], dtype=object)
```

[19]:
```
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
```

[20]:
```
model.feature_importances_
```

[20]:
```
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., 0., 0., 0., 0.])
```

[21]:
```
model.feature_importances_.argmax()
```

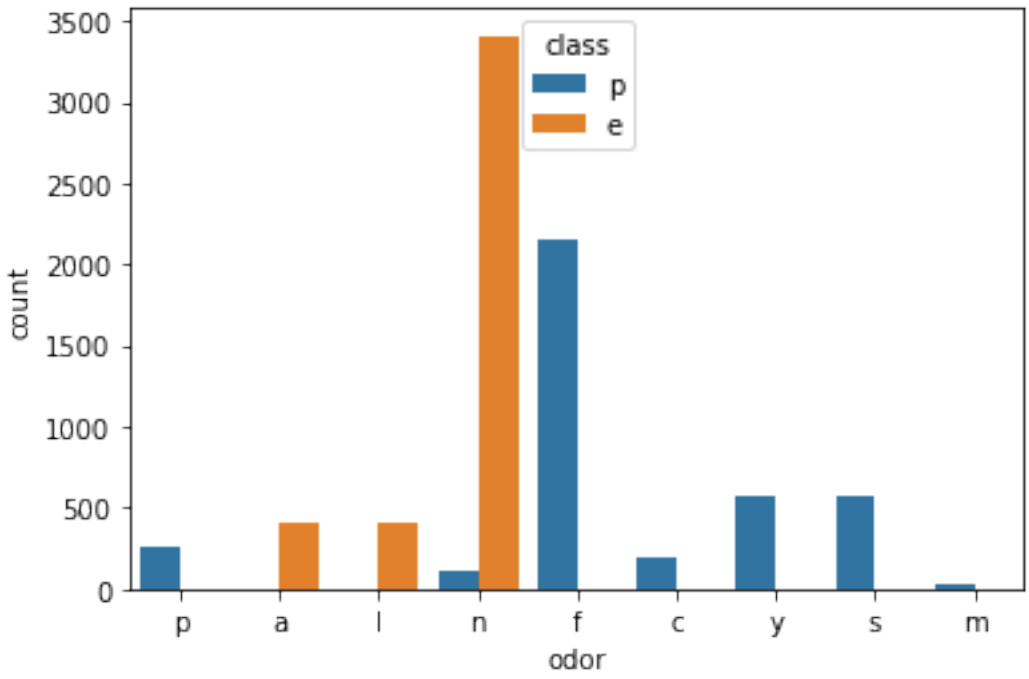
[21]:
```
22
```

[22]:
```
X.columns[22]
```

[22]:
```
'odor_n'
```

[23]:
```
sns.countplot(data=df,x='odor',hue='class')
```

[23]:
```
<AxesSubplot:xlabel='odor', ylabel='count'>
```



5.2 Analyzing performance as more weak learners are added.

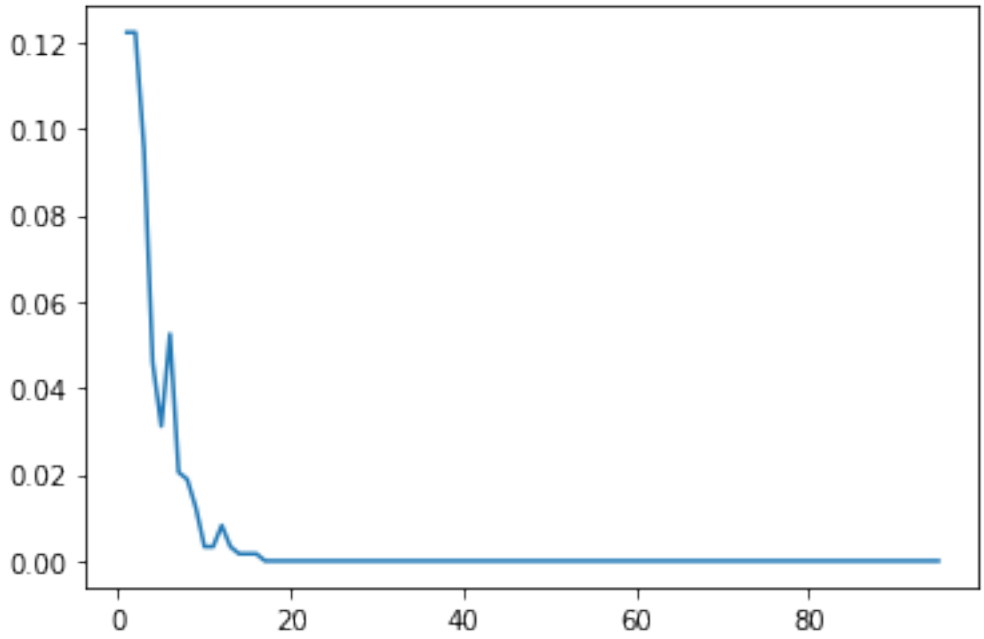
[24]:
```
len(X.columns)
```

[24]: 95

[25]:
```
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)
```

[26]:
```
plt.plot(range(1,96),error_rates)
```

[26]: [<matplotlib.lines.Line2D at 0x289c33b1f70>]



[27]:
```
model
```

[27]:
```
AdaBoostClassifier(n_estimators=95)
```

[28]:
```
model.feature_importances_
```

[28]:
```
array([0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.01052632, 0.        ,
       0.        , 0.01052632, 0.        , 0.        , 0.        ,
       0.01052632, 0.        , 0.05263158, 0.03157895, 0.03157895,
       0.        , 0.        , 0.06315789, 0.02105263, 0.        ,
       0.        , 0.        , 0.09473684, 0.09473684, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.01052632, 0.01052632, 0.        , 0.        , 0.        ,
       0.06315789, 0.        , 0.        , 0.        , 0.        ,
       0.03157895, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.06315789, 0.        , 0.        ,
       0.01052632, 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.01052632, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.05263158, 0.        , 0.16842105, 0.        , 0.10526316,
       0.        , 0.        , 0.04210526, 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.01052632])
```

[26]:
```
feats = pd.DataFrame(index=X.columns,data=model.
↪feature_importances_,columns=['Importance'])
```

[27]:
```
feats
```

[27]:

	Importance
cap-shape_c	0.000000
cap-shape_f	0.000000
cap-shape_k	0.000000
cap-shape_s	0.000000
cap-shape_x	0.000000
…	…
habitat_l	0.000000
habitat_m	0.000000
habitat_p	0.000000
habitat_u	0.000000
habitat_w	0.010526

[95 rows x 1 columns]

[28]:
```
imp_feats = feats[feats['Importance']>0]
```

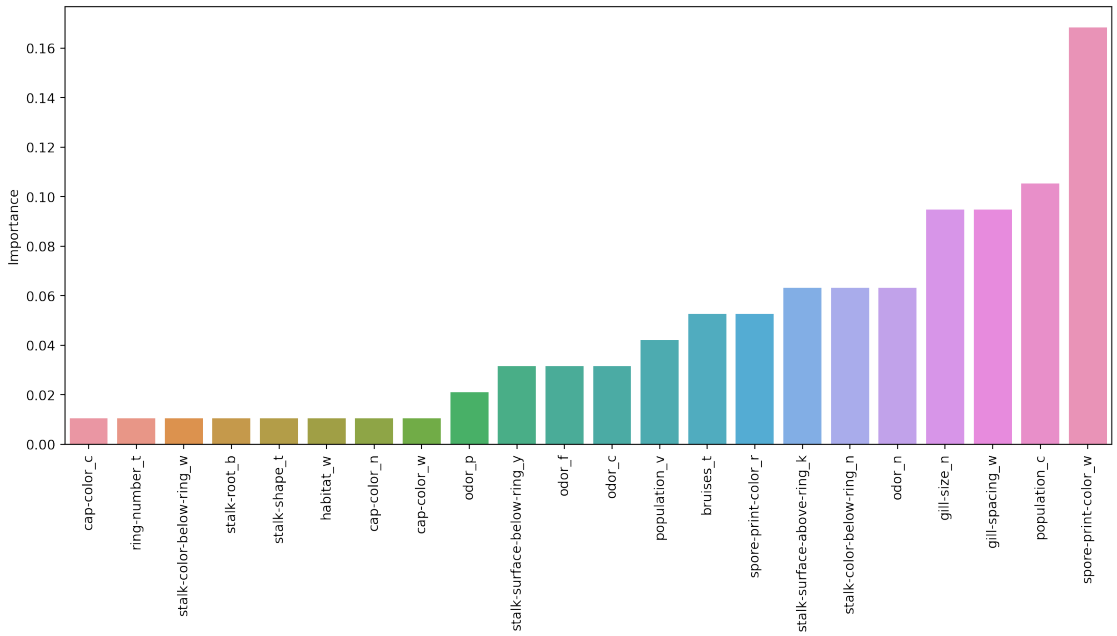
[31]:
```
imp_feats
```

[31]:

	Importance
cap-color_c	0.010526
cap-color_n	0.010526
cap-color_w	0.010526
bruises_t	0.052632
odor_c	0.031579
odor_f	0.031579
odor_n	0.063158
odor_p	0.021053
gill-spacing_w	0.094737
gill-size_n	0.094737
stalk-shape_t	0.010526
stalk-root_b	0.010526
stalk-surface-above-ring_k	0.063158
stalk-surface-below-ring_y	0.031579
stalk-color-below-ring_n	0.063158
stalk-color-below-ring_w	0.010526
ring-number_t	0.010526
spore-print-color_r	0.052632
spore-print-color_w	0.168421
population_c	0.105263
population_v	0.042105
habitat_w	0.010526

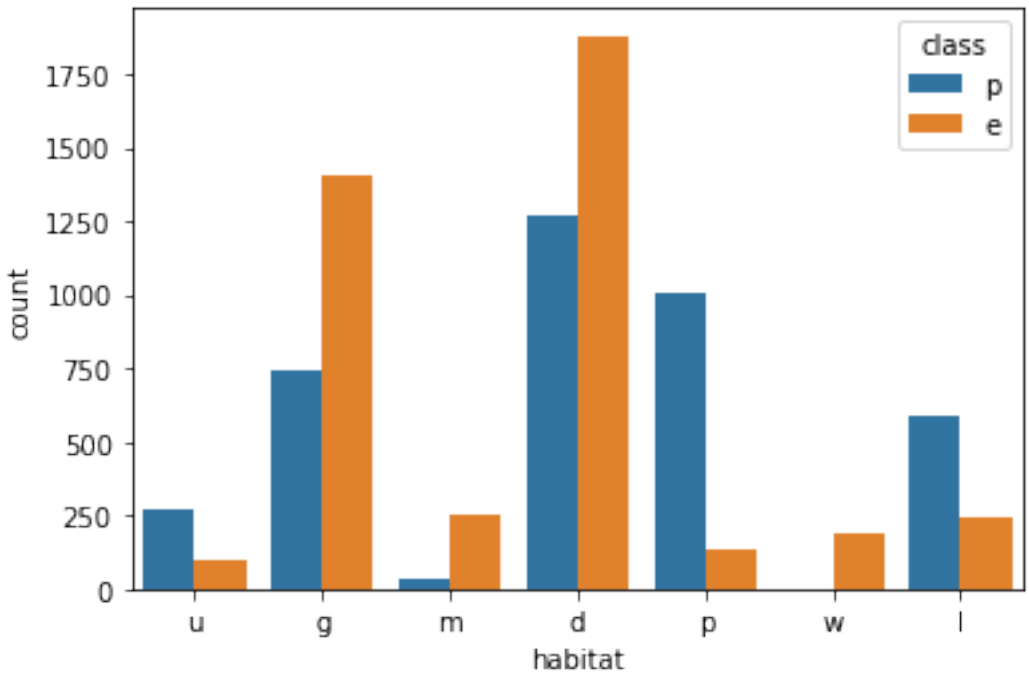
[32]:
```
imp_feats = imp_feats.sort_values("Importance")
```

[33]:
```
plt.figure(figsize=(14,6),dpi=200)
sns.barplot(data=imp_feats.sort_values('Importance'),x=imp_feats.
 ↪sort_values('Importance').index,y='Importance')

plt.xticks(rotation=90);
```



[34]:
```
sns.countplot(data=df,x='habitat',hue='class')
```

[34]:
```
<AxesSubplot:xlabel='habitat', ylabel='count'>
```



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!

Interessante ver como a importância dos recursos muda à medida que mais podem ser adicionados! Mas lembre-se de que todos esses são tocos de aprendizado fracos e a importância do recurso está disponível para todos os métodos de árvore!

5.3 Conclusion

In this project, we utilized the AdaBoost algorithm to create a predictive model for identifying the edibility of mushrooms based on various features. Our objective went beyond building a typical predictive model; we aimed to establish a general guideline for mushroom hunters by harnessing the power of machine learning and boosting.

We started by analyzing a dataset sourced from the UCI Machine Learning Repository, which included descriptions of hypothetical samples of gilled mushrooms. The dataset provided information on features such as cap shape, odor, gill spacing, and spore print color, among others. We recognized that determining the edibility of mushrooms is not a straightforward task and lacks a simple rule like “leaflets three” for Poisonous Oak and Ivy.

After preprocessing the data and encoding categorical features, we split it into training and testing sets. Next, we constructed an AdaBoostClassifier and trained it on the training data. We evaluated the model’s performance using metrics such as accuracy, precision, recall, and F1-score.

The results showed that our AdaBoost model achieved an accuracy of 88% in classifying edible and poisonous mushrooms. The model demonstrated good precision and recall for both classes, indi-

cating its effectiveness in identifying edible mushrooms while minimizing the risk of misclassifying poisonous ones.

We also investigated the importance of features in the AdaBoost model. By analyzing the feature importances, we identified key characteristics to consider when picking mushrooms. Features such as odor, gill spacing, gill size, spore print color, and population appeared to be significant factors in determining the edibility of mushrooms. These findings can serve as a useful guideline for mushroom hunters in assessing the safety of their harvest.

It's important to note that the model's performance and feature importances were based on weak learner stumps, highlighting the iterative nature of AdaBoost and its ability to leverage multiple weak learners to create a strong ensemble model.

Overall, our project demonstrated the potential of machine learning and boosting techniques in providing valuable insights and guidelines for real-world applications. By leveraging the power of data and predictive modeling, we aim to contribute to the safety and awareness of mushroom hunting enthusiasts.

5.4 Conclusão

Neste projeto, utilizamos o algoritmo AdaBoost para criar um modelo preditivo capaz de identificar a comestibilidade de cogumelos com base em diferentes características. Nosso objetivo foi além de construir um modelo preditivo comum; buscamos estabelecer uma diretriz geral para os caçadores de cogumelos, aproveitando o poder do aprendizado de máquina e do boosting.

Iniciamos analisando um conjunto de dados obtido do repositório UCI Machine Learning, que continha descrições de amostras hipotéticas de cogumelos com lamelas. O conjunto de dados fornecia informações sobre características como formato do chapéu, odor, espaçamento das lamelas e cor da impressão de esporos, entre outras. Reconhecemos que determinar a comestibilidade de cogumelos não é uma tarefa simples e não possui uma regra clara como "folhetos três" para o carvalho venenoso e a hera venenosa.

Após o pré-processamento dos dados e a codificação das características categóricas, dividimos o conjunto de dados em conjuntos de treinamento e teste. Em seguida, construímos um AdaBoostClassifier e o treinamos com os dados de treinamento. Avaliamos o desempenho do modelo utilizando métricas como acurácia, precisão, recall e pontuação F1.

Os resultados mostraram que nosso modelo AdaBoost alcançou uma acurácia de 88% na classificação de cogumelos comestíveis e venenosos. O modelo apresentou boa precisão e recall para ambas as classes, indicando sua eficácia na identificação de cogumelos comestíveis, ao mesmo tempo em que minimiza o risco de classificar erroneamente cogumelos venenosos.

Também investigamos a importância das características no modelo AdaBoost. Ao analisar a importância das características, identificamos características-chave a serem consideradas ao colher cogumelos. Características como odor, espaçamento das lamelas, tamanho das lamelas, cor da impressão de esporos e população pareceram ser fatores significativos na determinação da comestibilidade dos cogumelos. Essas descobertas podem servir como uma diretriz útil para os caçadores de cogumelos ao avaliar a segurança de sua colheita.

É importante ressaltar que o desempenho do modelo e a importância das características foram baseados em classificadores fracos, evidenciando a natureza iterativa do AdaBoost e sua capacidade de aproveitar múltiplos classificadores fracos para criar um modelo de conjunto forte.

Em suma, nosso projeto demonstrou o potencial das técnicas de aprendizado de máquina e boosting ao fornecer insights valiosos e diretrizes para aplicações do mundo real. Ao aproveitar o poder dos dados e da modelagem preditiva, buscamos contribuir para a segurança e conscientização dos entusiastas da caça aos cogumelos.

5.5 End

5.6 Fim