BIRMINGHAM CITY
University
Bl

CMP6202
AI and Machine Learning Project
2022–2023

Individual Report

Mushroom Classification

BSc Computer Science
Bianca Morais
20128324

Table of Contents

1 Report Introduction ... 3
1.1 Dataset identification ... 3
1.2 Supervised learning task identification ... 7
1.3 Team Identification ... 7
2 Exploratory Data Analysis ... 8
2.1 Question(s) identification ... 8
2.2 Splitting the dataset ... 9
2.3 Exploratory Data Analysis process and results ... 10
2.4 EDA conclusions ... 28
3 Experimental Design ... 28
3.1 Identification of your chosen supervised learning algorithm(s) ... 28
3.2 Identification of appropriate evaluation techniques ... 29
3.3 Data cleaning and Pre-processing transformations ... 30
3.4 Limitations and Options ... 30
4 Predictive Modelling / Model Development ... 31
4.1 The predictive modelling process ... 31
4.2 Evaluation results on “seen” data ... 32
5 Evaluation and further modelling improvements ... 33
5.1 Initial evaluation comparison ... 33
5.2 Final Evaluation results ... 35
6 Conclusion ... 36
6.1 Summary of results ... 36
6.2 Suggested further improvements to the model development process ... 36
6.3 Reflection on Research Team ... 36
6.4 Reflection on Individual Learning ... 36
7 References ... 37

1 Report Introduction

This report focuses on developing a supervised machine learning algorithm able to determine if a mushroom is poisonous based on attributes such as cap shape, colour, smell and gills. My team and I have each chosen a different algorithm to attempt to classify the same dataset of mushrooms. Being used are the following; logistic regression which provides a yes or no result based on given variables, support vector machine is a graph based analysis which draws a sort of line of best fit, a decision tree which will categorise mushrooms based on certain questions, somewhat like a flowchart and a random forest which uses multiple decision trees to classify the mushrooms

1.1 Dataset identification

The dataset, titled Mushroom Classification contains the attributes of 23 different species of mushrooms and features 23 columns and 8125 rows. While found on Kaggle, the original dataset was hosted on the UCI Machine Learning Repository. Each row represents a different mushroom with 23 different attributes listed on the next page. The dataset was gathered from shrooming enjoyers. Specifically, the mushrooms found belong to the Agaricus and Lepoita families.

The dataset details the following attributes:

Attribute name	Values	Model Values	Description

Class	edible=e
	poisonous=p
		1
		0
			Is the mushroom edible or poisonous?

Cap shape	bell=b
	conical=c,
	convex=x,
	flat=f,
	knobbed=k,
	sunken=s
		b
		c
		x
		f
		l
		s
			The cap of the mushroom is the topmost part and gives the umbrella shape

Cap surface	fibrous=f,
	grooves=g,
	scaly=y,
	smooth=s
		f
		g
		y
		s
			The texture of the topmost part of the mushroom

Cap colour	brown=n,
	buff=b,
	cinnamon=c,
	gray=g,
	green=r,
	pink=p,
	purple=u,
	red=e,
	white=w,
	yellow=y
		n
		b
		c
		g
		r
		p
		u
		e
		w
		y
			The colour of the topmost part of the mushroom

Bruises	bruises=t,
	no=f
		t
		f
			Nicking the top and bottom of the mushroom cap and observing any colour changes

Odour	almond=a,
	anise=l,
	creosote=c,
	fishy=y,
	foul=f,
	musty=m,
	none=n,
	pungent=p,
	spicy=s
		a
		l
		c
		y
		f
		m
		n
		p
		s
			What the mushroom smells like

Gill Attachment	attached=a,
	descending=d,
	free=f,
	notched=n
		a
		d
		f
		n
			The way in which the gills attach to the stem

Gill spacing	close=c,
	crowded=w,
	distant=d
		c
		w
		d
			The amount of space between each of the gills of the mushroom

Gill size	broad=b,
	narrow=n
		b
		n
			The size of the gill itself (wide or narrow)

Gill colour	black=k,
	brown=n,
		k
		n
			The colour of the gills themselves

	buff=b,
	chocolate=h,
	gray=g,
	green=r,
	orange=o,
	pink=p,
	purple=u,
	red=e,
	white=w,
	yellow=y
		b
		h
		g
		r
		o
		p
		u
		e
		w
		y	

Stalk shape	enlarging=e,
	tapering=t
		e
		t
			The direction the stalk widens in.

Stalk root	bulbous=b,
	club=c,
	cup=u,
	equal=e,
	rhizomorphs=z,
	rooted=r,
	missing=?
		b
		c
		u
		e
		z
		r
		?
			The shape of the stalk's root

Stalk surface above ring	fibrous=f,
	scaly=y,
	silky=k,
	smooth=s
		f
		y
		k
		s
			How the stalk feels and looks above the ring.

Stalk surface below ring	fibrous=f,
	scaly=y,
	silky=k,
	smooth=s
		f
		y
		k
		s
			The stalk's surface below the ring.

Stalk colour above ring	brown=n,
	buff=b,
	cinnamon=c,
	gray=g,
	orange=o,
	pink=p,
	red=e,
	white=w,
	yellow=y
		n
		b
		c
		g
		o
		p
		e
		w
		y
			The colour of the stalk above the ring.

Stalk colour below ring	brown=n,
	buff=b,
	cinnamon=c,
	gray=g,
	orange=o,
	pink=p,
	red=e,
	white=w,
	yellow=y
		n
		b
		c
		g
		o
		p
		e
		w
		y
			The colour of the stalk below the ring.

Veil type	partial=p,
	universal=u
		p
		u
			The mushroom's veil is a membrane that covers the cap and stalk in a young mushroom.

Veil colour	brown=n,
	orange=o,
	white=w,
	yellow=y
		n
		o
		w
		y
			The colour of the mushroom's veil.

Ring number	none=n,
	one=o,
	two=t
		n
		o
		t
			How many rings the mushroom has.

Ring type	cobwebby=c,
	evanescent=e,
	flaring=f,
	large=l,
	none=n,
	pendant=p,
	sheathing=s,
	zone=z
		c
		e
		f
		l
		n
		p
		s
		z
			The type of ring the mushroom has.

Spore print colour	black=k,
	brown=n,
	buff=b,
	chocolate=h,
	green=r,
	orange=o,
	purple=u,
	white=w,
	yellow=y
		k
		n
		b
		h
		r
		o
		u
		w
		y
			The powdery deposit obtained by dropping the mushroom’s spores onto a surface.

Population	abundant=a,
	clustered=c,
	numerous=n,
	scattered=s,
	several=v,
	solitary=y
		a
		c
		n
		s
		v
		y
			How many mushrooms are in one colony.

Habitat	grasses=g,
	leaves=l,
	meadows=m,
	paths=p,
	urban=u,
	waste=w,
	woods=d
		q
		l
		m
		p
		u
		w
		d
			Where the mushroom is found.

1.2 Supervised learning task identification

My team and I are going to attempt to create a set of machine learning models to identify whether or not a mushroom is edible or not based on varying attributes. Since the results we're aiming for is akin to a yes or no question, we've chosen algorithms capable of classification, as shown below in section 1.3.

1.3 Team Identification

Forename	**Surname**	**Student ID**	**Model(s) developed**
Bianca	Morais	20128324	Logistic Regression
Kristine	Denoveva	20131983	Decision Tree
Jack	Farmer	20134782	Random Forest
Daniel	Foster	19131813	Support Vector Machine

2 Exploratory Data Analysis

2.1 Question(s) identification

The dataset contains a variety of mushrooms that are either poisonous or edible. The aim is to see how likely the chance of a mushroom being safe to eat is based on a number of different attributes. Sub-questions can come from this one aim based on the different variables, such as what kind of gill, cap colour or location is most likely to be poisonous. This can be cut even smaller per value to identify if a mushroom with an almond odour, for example, is edible.

While on paper this seems definitive, in reality it’s much harder to tell a mushroom’s edibility without sufficient knowledge. Rules of thumb for mushroom edibility, such as peelable caps and if it grows on wood can be subverted such as the peelable cap of a death cap or how funeral bells are found on wood. Even mushrooms with traits of other toxic mushrooms can still be safe to eat. When you think of a mushroom, the most common one to come to mind is the red capped, white gilled amanita muscaria. While it looks dangerous with its red cap and white gills, both common indicators of a toxic mushroom, the amanita itself is completely safe to eat while only having hallucinogenic properties.

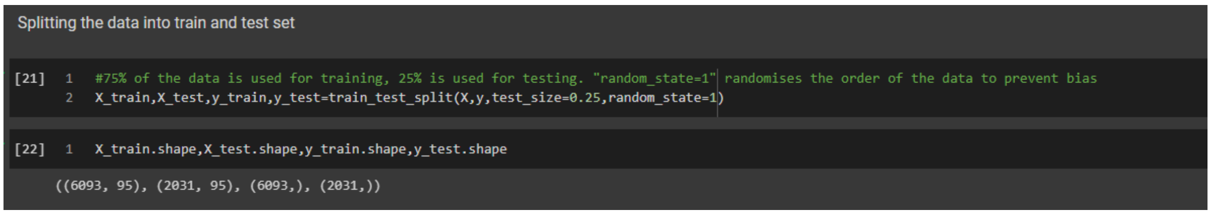
This is why it’s important to state that even if the EDA proves to be correct 100% of the time, it still shouldn’t be taken fully at face value.

Amanita Muscaria image sourced from Wikipedia

2.2 Splitting the dataset

75% of the dataset will be randomly selected for training, while the remaining 25% will be used to test the model. No validation sets will be used. In order to avoid any data leakage from occurring, the training and testing tables will be kept separately in different files. By keeping them in separate files, we also guarantee that our testing and training data will remain constant among us for easy comparison.

Additionally, the testing and training tables with be split into separate files titles X and Y to keep the edibility separate. X will contain the 22 other columns while Y will only contain the edibility.



Splitting the data into train and test set

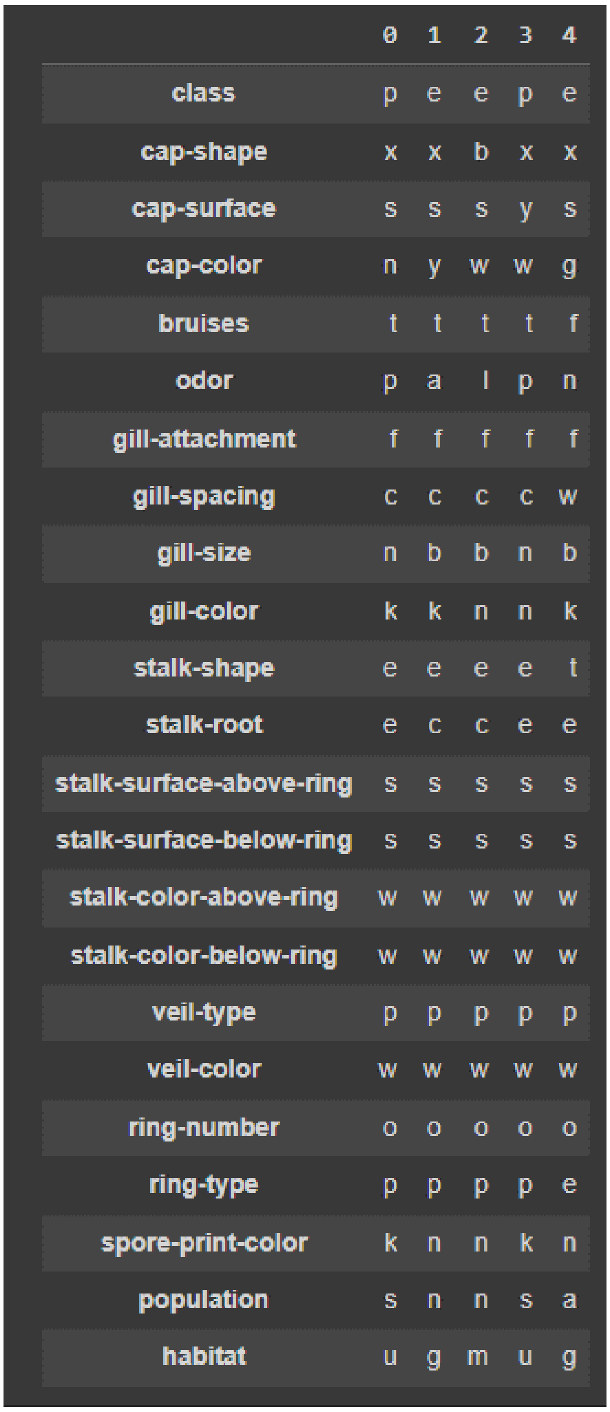
```
[21]  #75% of the data is used for training, 25% is used for testing. "random_state=1" randomises the order of the data to prevent bias
      X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=1)

[22]  X_train.shape,X_test.shape,y_train.shape,y_test.shape

      ((6093, 95), (2031, 95), (6093,), (2031,))
```

2.3 Exploratory Data Analysis process and results

First, we load the mushroom dataset with:

```
dfmushroom = pd.read_csv('mushrooms.csv')
dfmushroom.head().T
```

This provides us with a table featuring every single row of data in the dataset.

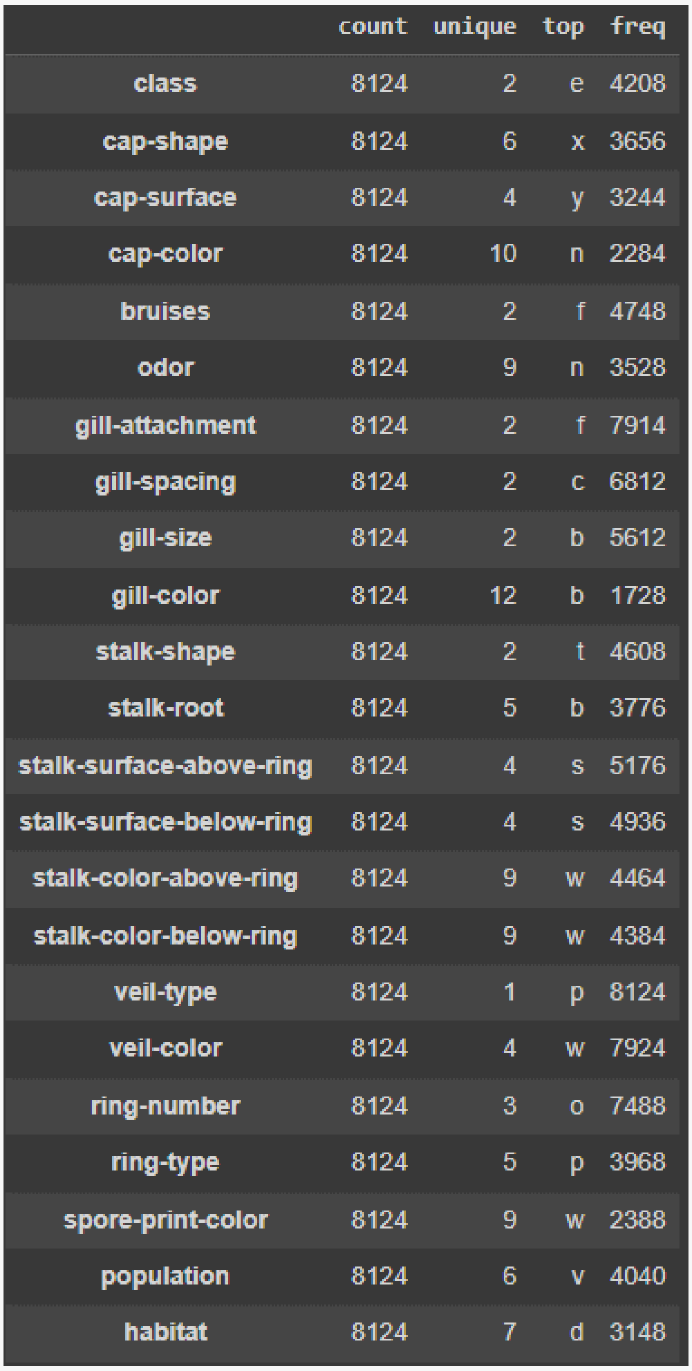


	0	1	2	3	4
class	p	e	e	p	e
cap-shape	x	x	b	x	x
cap-surface	s	s	s	y	s
cap-color	n	y	w	w	g
bruises	t	t	t	t	f
odor	p	a	l	p	n
gill-attachment	f	f	f	f	f
gill-spacing	c	c	c	c	w
gill-size	n	b	b	n	b
gill-color	k	k	n	n	k
stalk-shape	e	e	e	e	t
stalk-root	e	c	c	e	e
stalk-surface-above-ring	s	s	s	s	s
stalk-surface-below-ring	s	s	s	s	s
stalk-color-above-ring	w	w	w	w	w
stalk-color-below-ring	w	w	w	w	w
veil-type	p	p	p	p	p
veil-color	w	w	w	w	w
ring-number	o	o	o	o	o
ring-type	p	p	p	p	e
spore-print-color	k	n	n	k	n
population	s	n	n	s	a
habitat	u	g	m	u	g

By running

```
dfmushroom.describe().T
```

we are provided with another table to review the different attributes our dataset has in a nonspecific, condensed way. The T is added on to set the columns to be vertical and easier to screenshot.



	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

This runs through how many entries we have and if any missing data is present, how many different variations of values for each row, the most common value for each one and how commonly they show up.

If we want to see the spread different values in an easier to read form, we can run this following line of code to present it in a text-based format. This will show exactly how many mushrooms have which property.

```
for i in dfmushroom.columns:
    dfmushroom[i] = dfmushroom[i].astype('category')
    print(dfmushroom[i].value_counts())
    print('='*50)
```

e   4208
p   3916
Name: class, dtype: int64

There’s an almost even split of mushrooms that are both edible and poisonous. One of my worries when it came to randomly splitting our table was that we would end up with an imbalance and potentially lead to skewed results, but with an even split of both, I think our models should provide accurate results.

x   3656
f   3152
k   828
b   452
s   32
c   4
Name: cap-shape, dtype: int64

y   3244
s   2556
f   2320
g   4
Name: cap-surface, dtype: int64

n   2284
g   1840
e   1500
y   1072
w   1040
b   168
p   144
c   44
r   16
u   16
Name: cap-color, dtype: int64

f   4748
t   3376
Name: bruises, dtype: int64

n   3528
f   2160
s   576
y   576

```
a     400
l    400
p     256
c     192
m      36
Name: odor, dtype: int64
```

```
f    7914
a     210
Name: gill-attachment, dtype: int64
```

```
c    6812
w    1312
Name: gill-spacing, dtype: int64
```

```
b    5612
n    2512
Name: gill-size, dtype: int64
```

```
b    1728
p    1492
w    1202
n    1048
g     752
h     732
u     492
k     408
e      96
y      86
o      64
r      24
Name: gill-color, dtype: int64
```

```
t    4608
e    3516
Name: stalk-shape, dtype: int64
```

```
b    3776
?    2480
e    1120
c     556
r     192
Name: stalk-root, dtype: int64
```

```
s    5176
k    2372
f     552
y      24
```

```
Name: stalk-surface-above-ring, dtype: int64
```

```
s    4936
k    2304
f     600
y     284
Name: stalk-surface-below-ring, dtype: int64
```

```
w    4464
p    1872
g     576
n     448
b     432
o     192
e      96
c      36
y       8
Name: stalk-color-above-ring, dtype: int64
```

```
w    4384
p    1872
g     576
n     512
b     432
o     192
e      96
c      36
y      24
Name: stalk-color-below-ring, dtype: int64
```

```
p    8124
Name: veil-type, dtype: int64
```

None of the mushrooms surveyed seemed to have a universal veil, making this almost irrelevant to determining the toxicity of a mushroom.

```
w    7924
n      96
o      96
y       8
Name: veil-color, dtype: int64
```

```
o    7488
t     600
n      36
Name: ring-number, dtype: int64
p    3968
e    2776
l    1296
f      48
```

```
n      36
Name: ring-type, dtype: int64
```

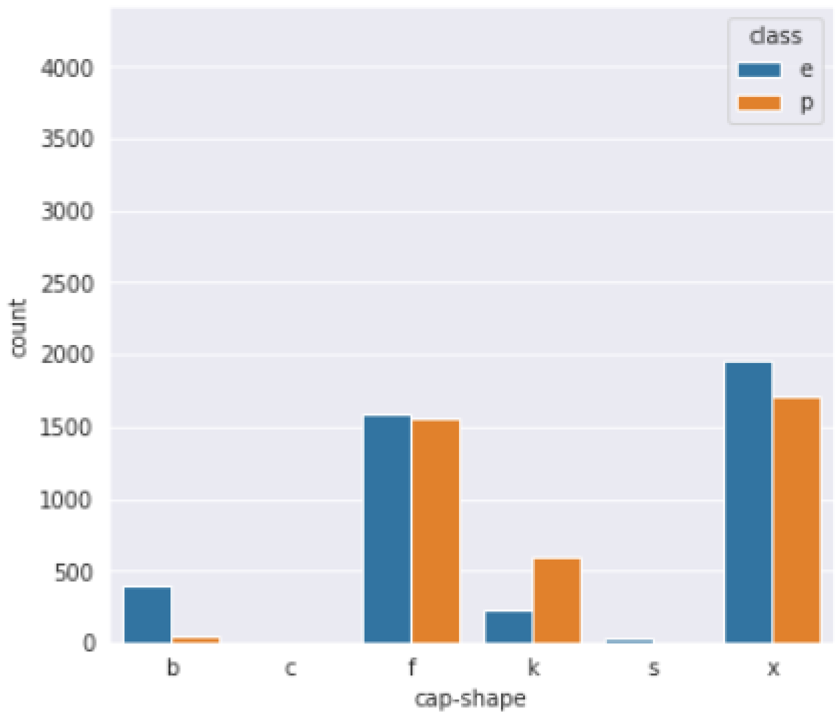
```
w    2388
n    1968
k    1872
h    1632
r      72
b      48
o      48
u      48
y      48
Name: spore-print-color, dtype: int64
```

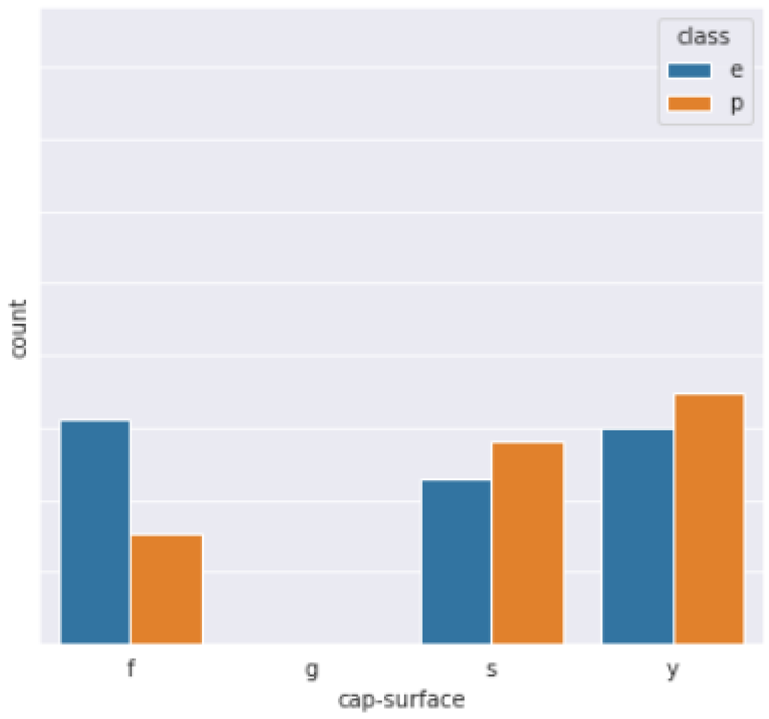
```
v    4040
y    1712
s    1248
n     400
a     384
c     340
Name: population, dtype: int64
```

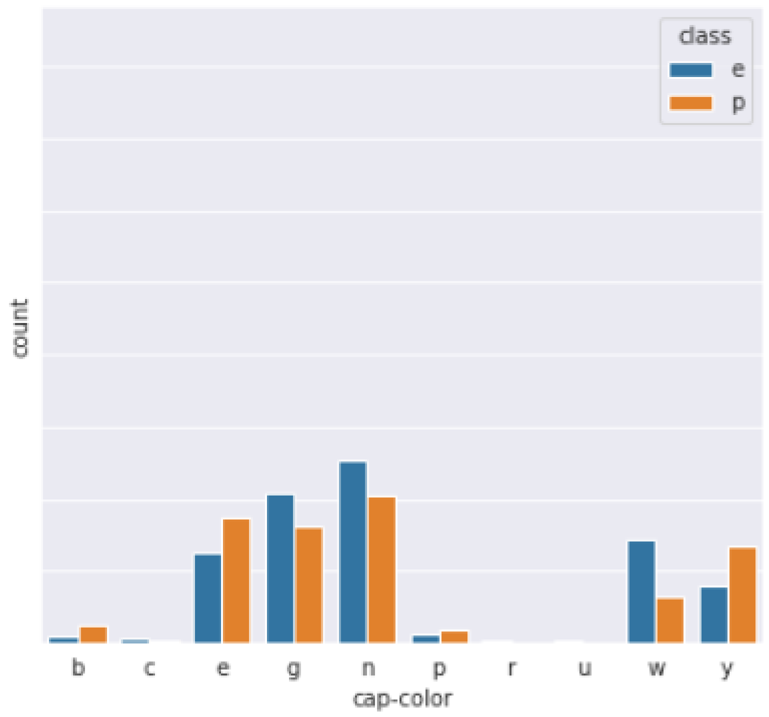
```
d    3148
g    2148
p    1144
l     832
u     368
m     292
w     192
Name: habitat, dtype: int64
```

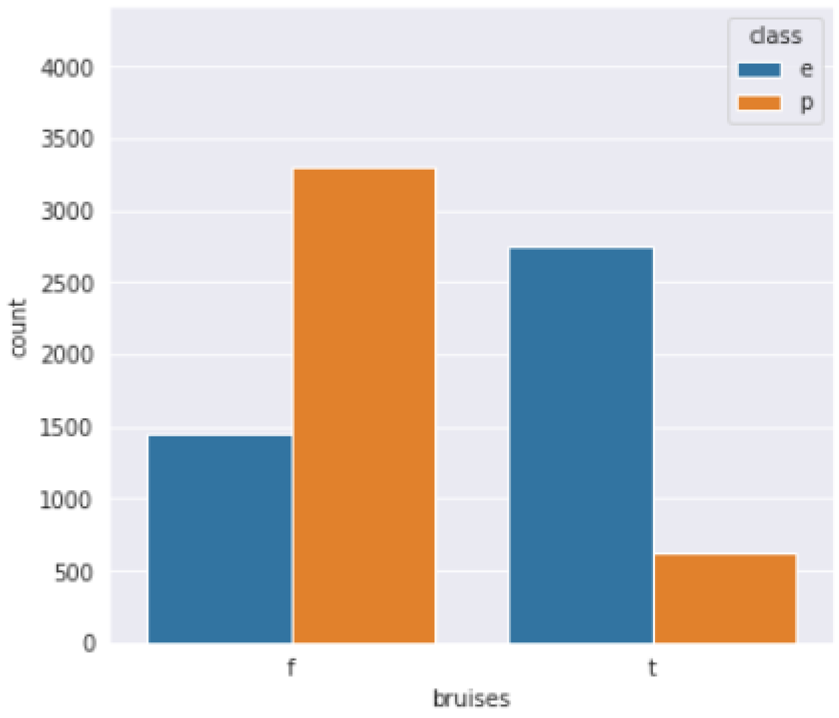
Using bar charts can also be a helpful way of visualising data at a glance. Since our goal is to find which mushrooms are edible, we can refer back to these to see if our results are at least somewhat correct!

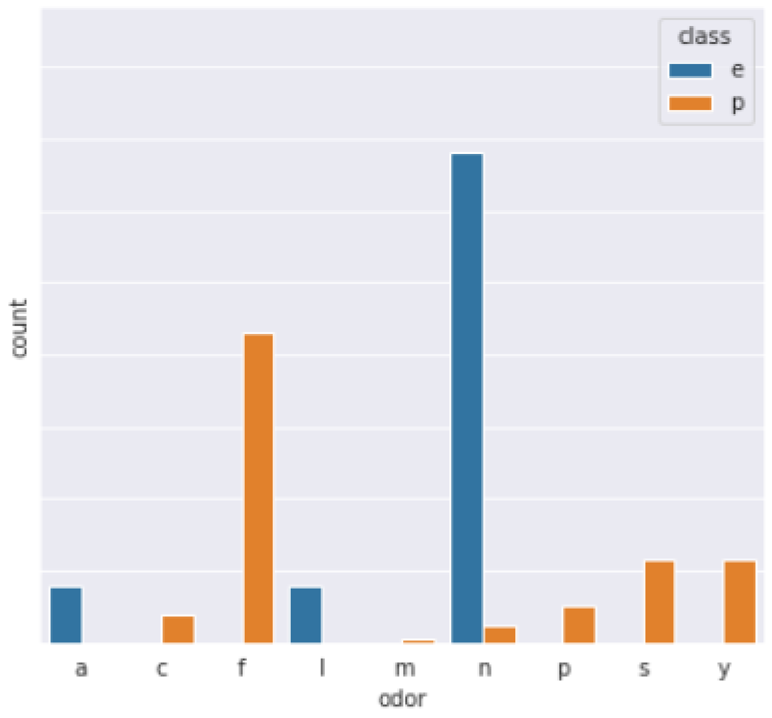
```
fig, axes = plt.subplots(nrows=7,ncols=3, figsize=(20,50),sharey=True)
idx = 0
for col in dfmushroom.columns[1:]:
    sns.countplot(data=dfmushroom,x=col, hue='class' ,ax=axes[idx//3][idx%3])
    idx += 1
```

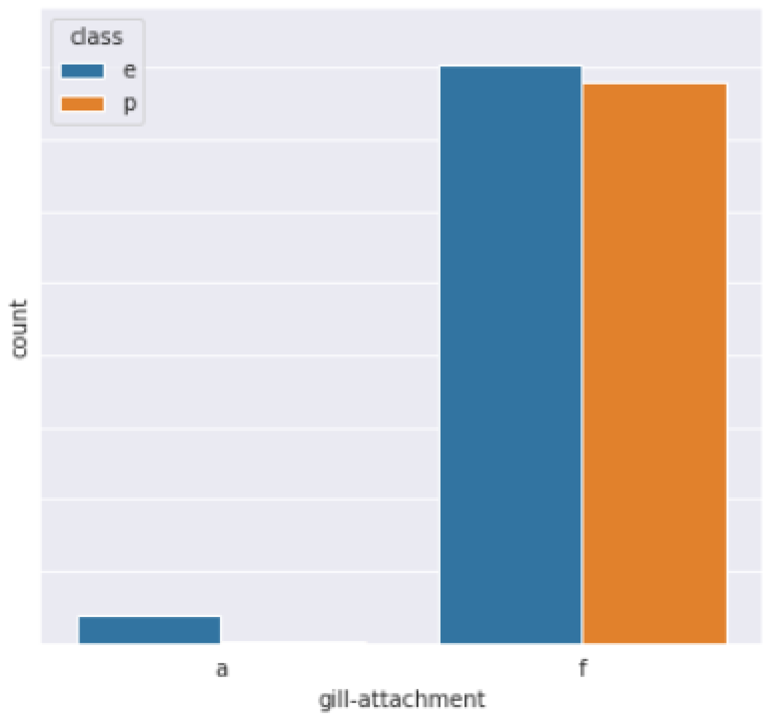

class
e
p
4000
3500
3000
2500
2000
1500
1000
500
0
count
b
c
f
k
s
x
cap-shape

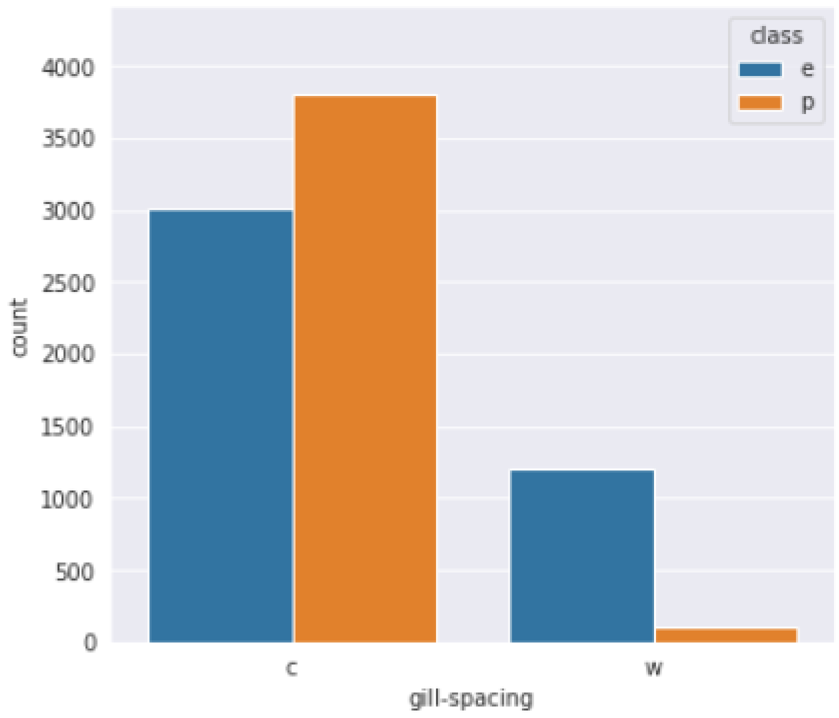

class
e
p
count
f
g
s
y
cap-surface

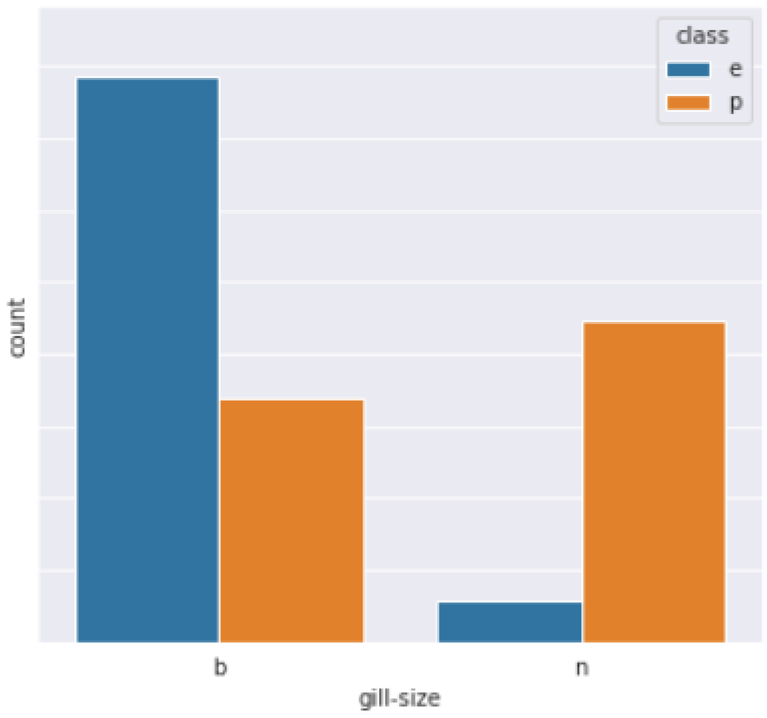

class
e
p
count
b
c
e
g
n
p
r
u
w
y
cap-color

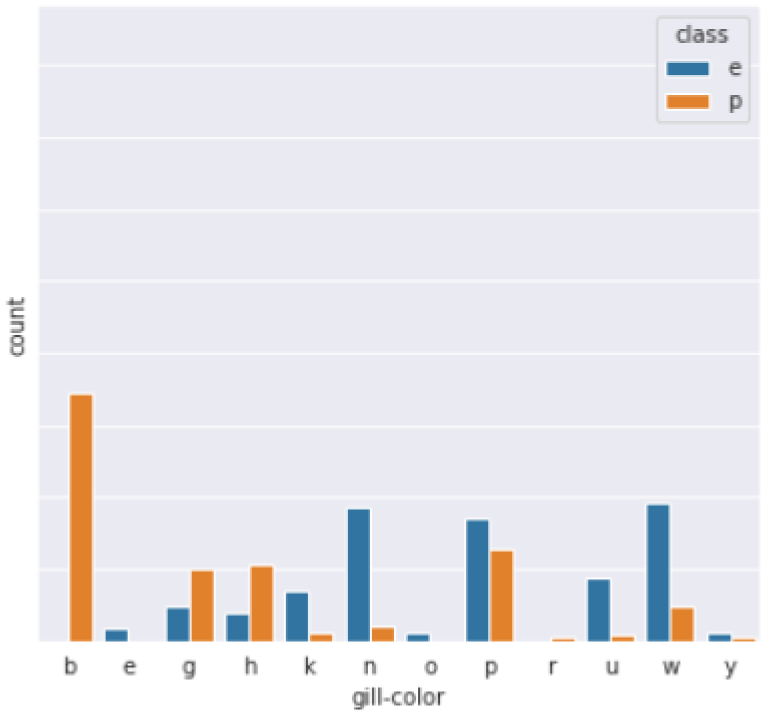

class
e
p
4000
3500
3000
2500
2000
1500
1000
500
0
count
f
t
bruises

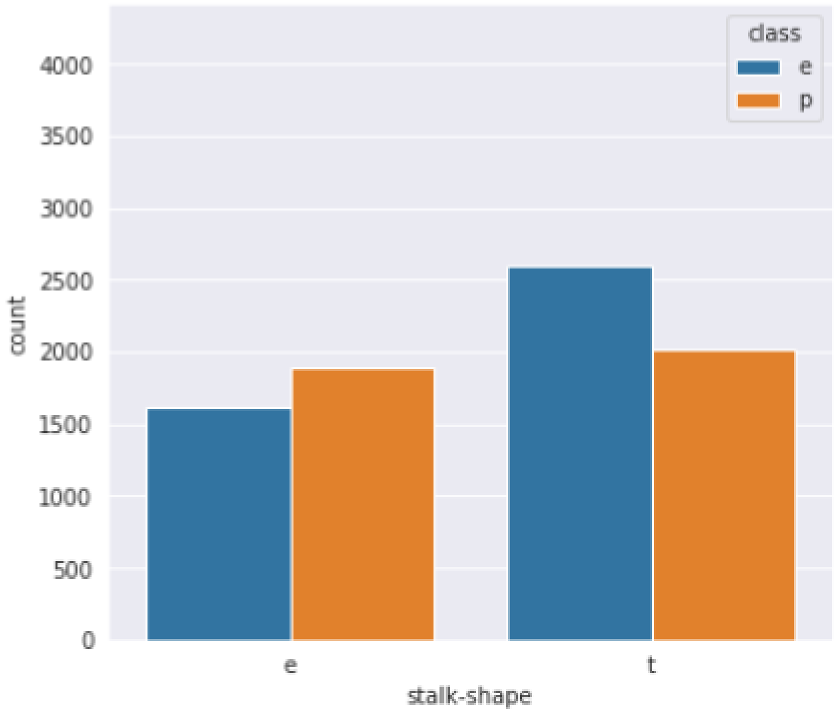

class
e
p
count
a
c
f
l
m
n
p
s
y
odor

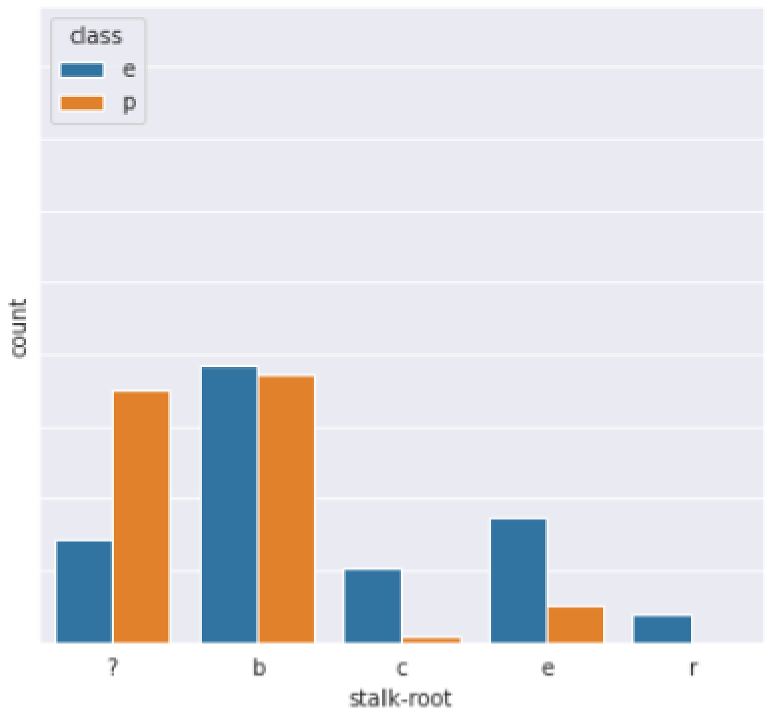

class
e
p
count
a
f
gill-attachment

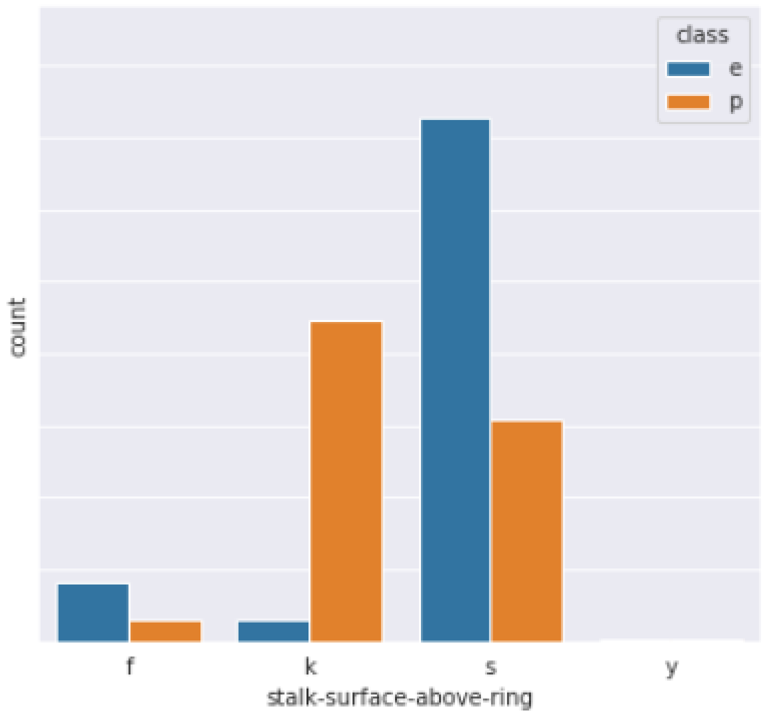

class
e
p
count
4000
3500
3000
2500
2000
1500
1000
500
0
c
w
gill-spacing

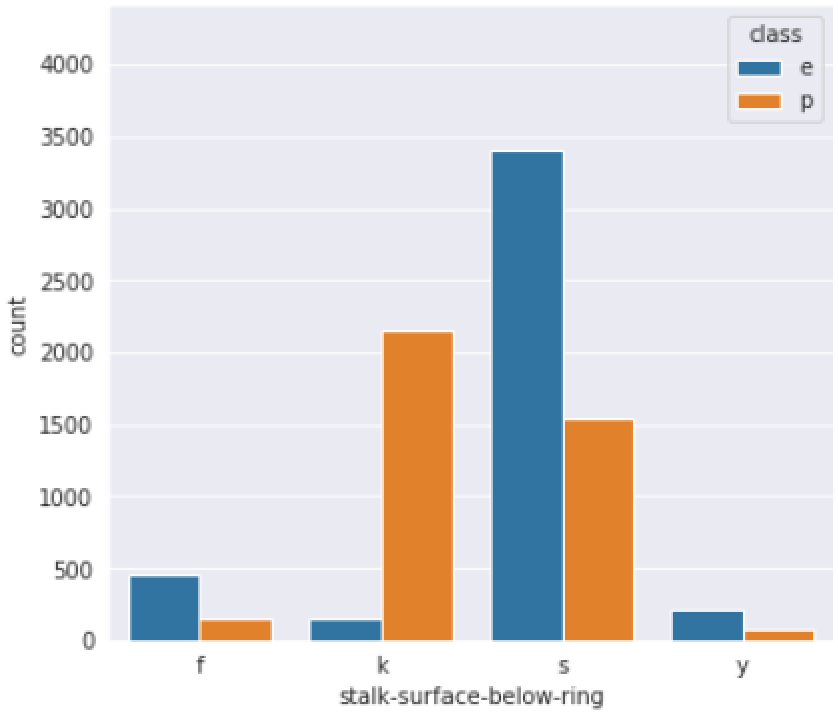

class
e
p
count
b
n
gill-size

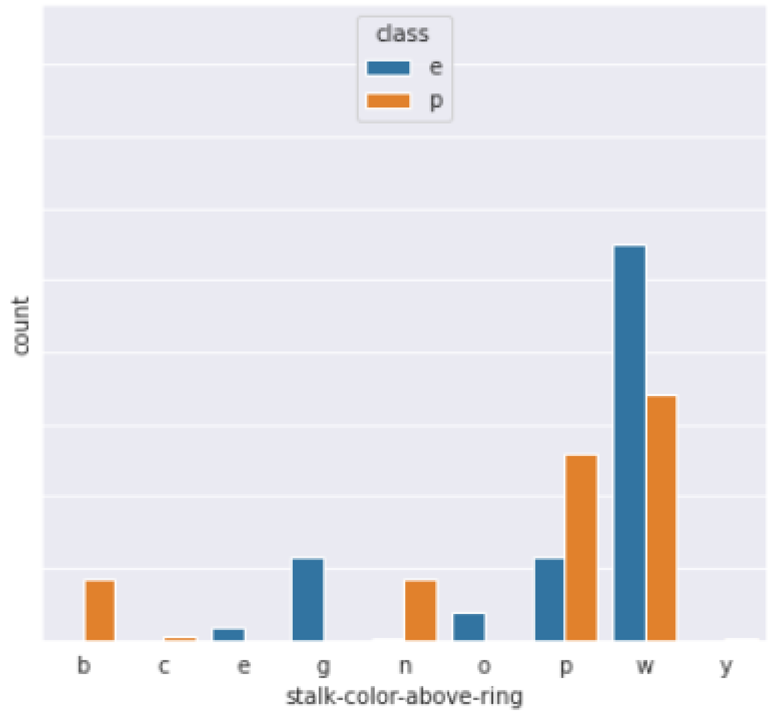

class
e
p
count
b
e
g
h
k
n
o
p
r
u
w
y
gill-color

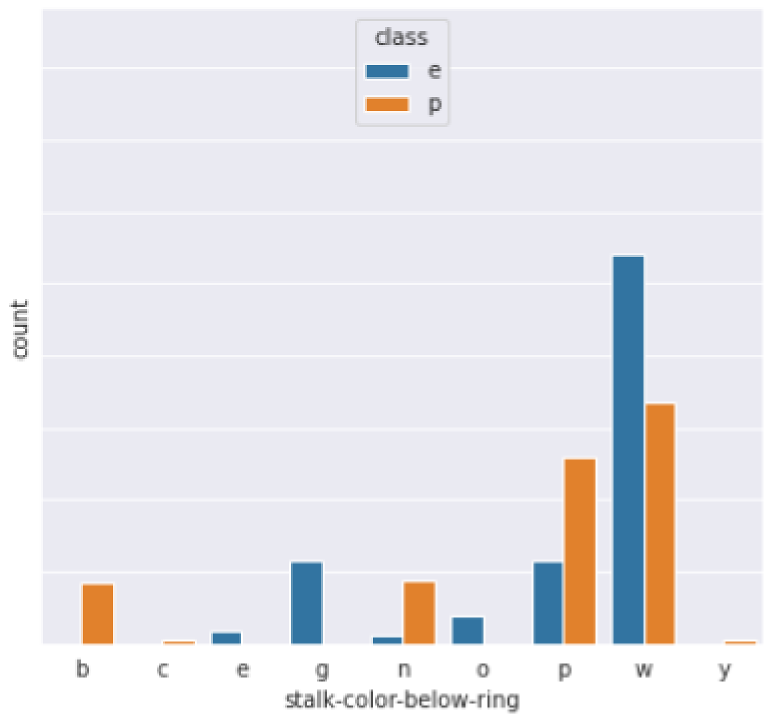

class
e
p
4000
3500
3000
2500
2000
1500
1000
500
0
count
e
t
stalk-shape

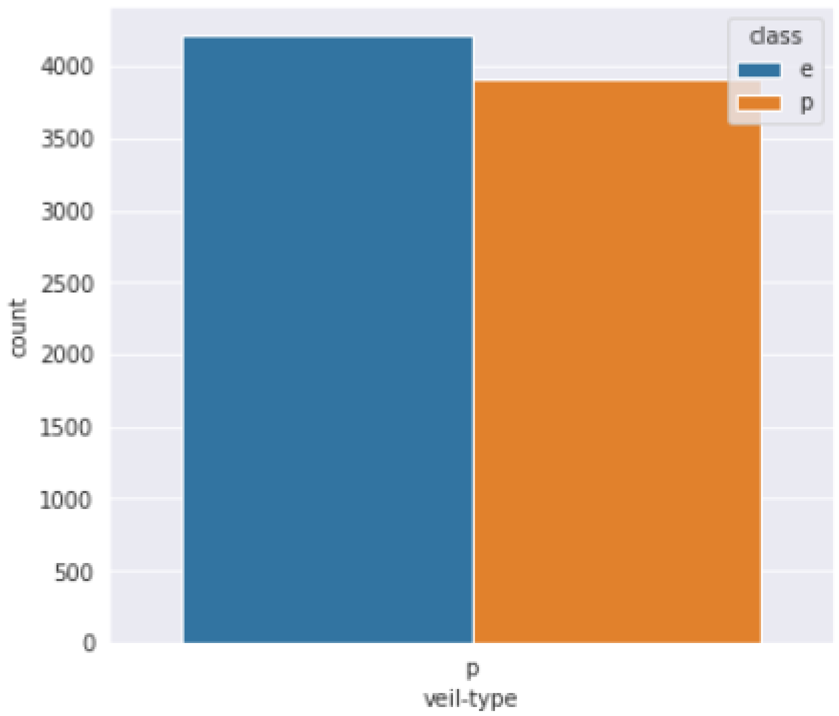

class
e
p
count
?
b
c
e
r
stalk-root

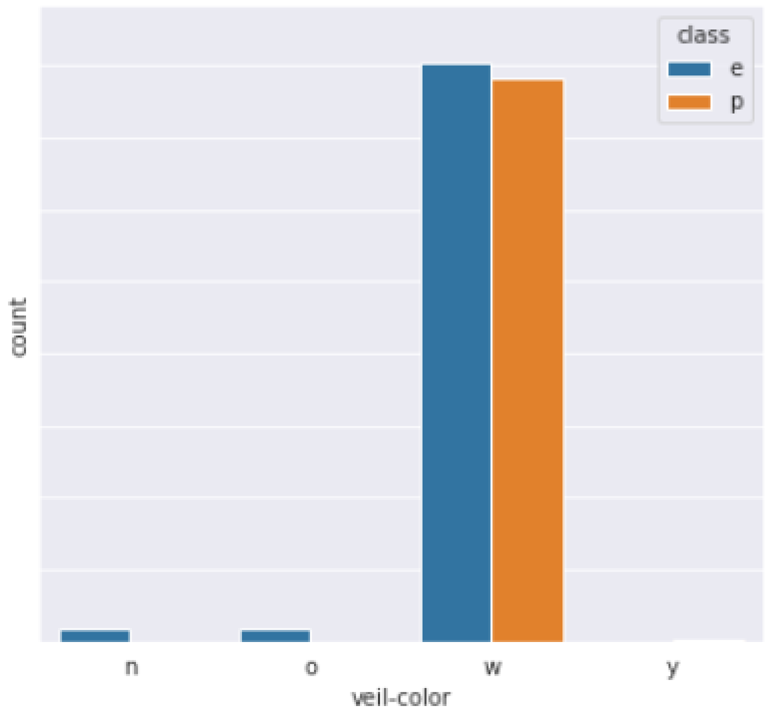

class
e
p
count
f
k
s
y
stalk-surface-above-ring

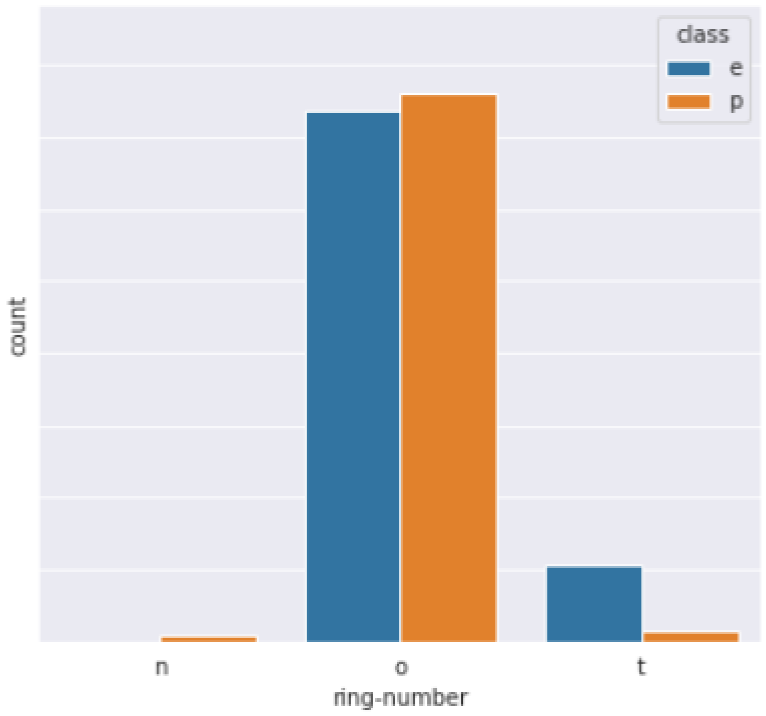

class
e
p
4000
3500
3000
2500
2000
1500
1000
500
0
count
f
k
s
y
stalk-surface-below-ring

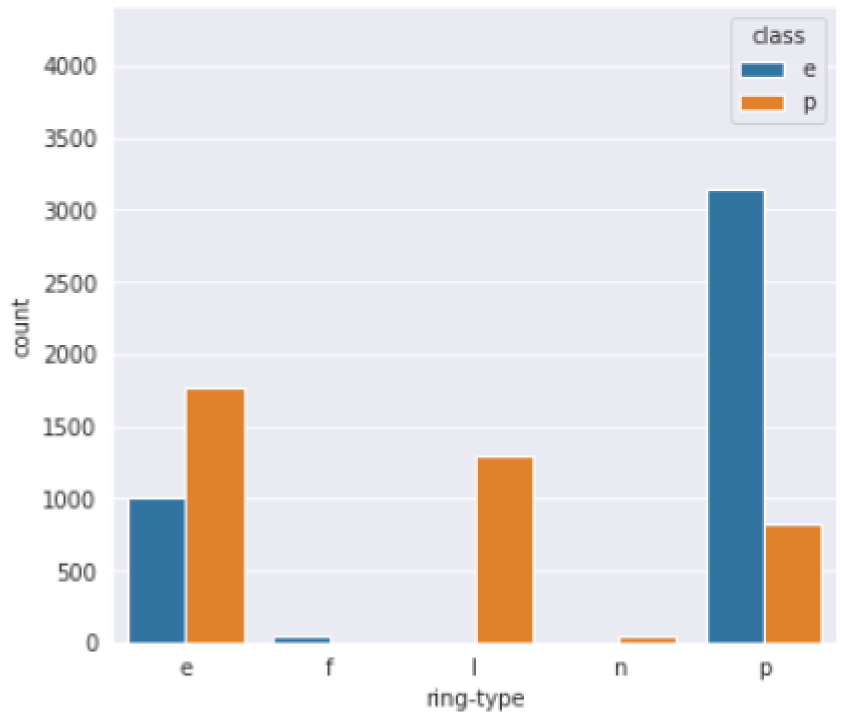

class
e
p
count
b
c
e
g
n
o
p
w
y
stalk-color-above-ring

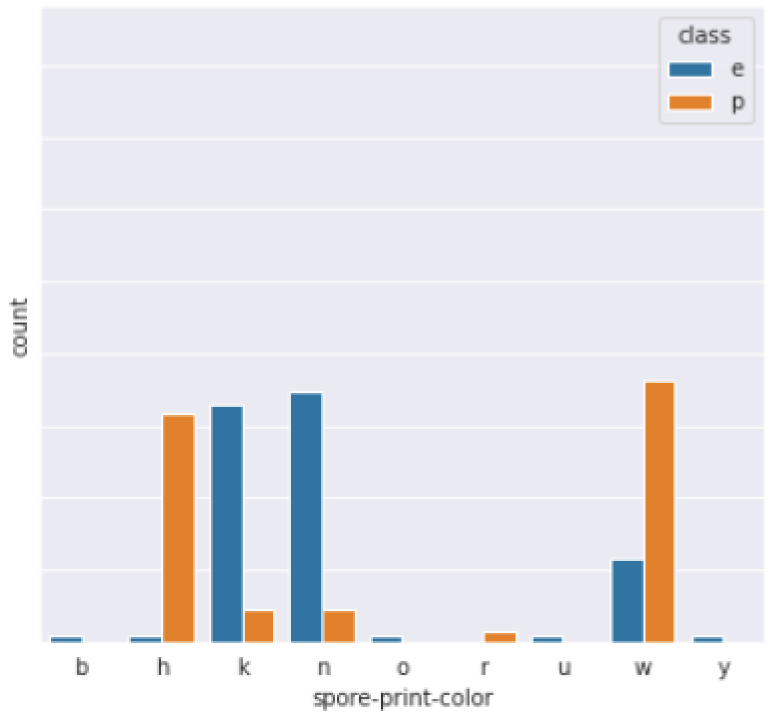

class
e
p
count
b
c
e
g
n
o
p
w
y
stalk-color-below-ring

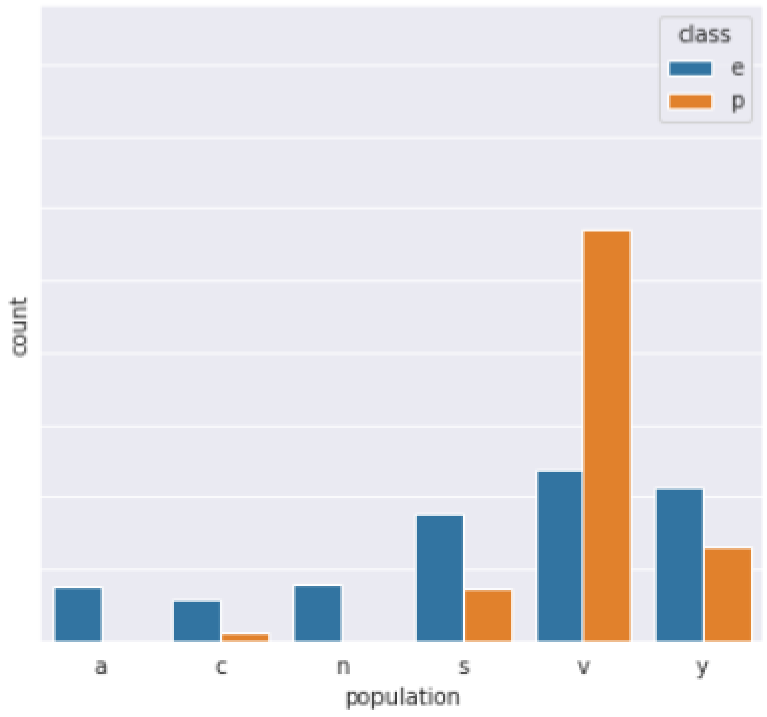

class
e
p
count
4000
3500
3000
2500
2000
1500
1000
500
0
p
veil-type

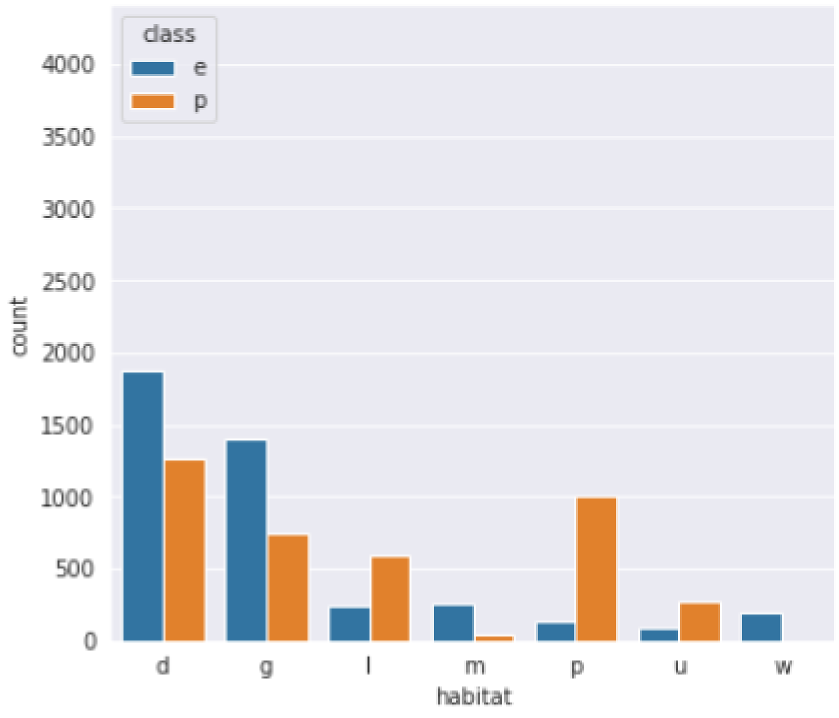

class
e
p
count
n
o
w
y
veil-color


class
e
p
count
n
o
t
ring-number


class
e
p
4000
3500
3000
2500
2000
1500
1000
500
0
count
e
f
l
n
p
ring-type


class
e
p
count
b
h
k
n
o
r
u
w
y
spore-print-color


class
e
p
count
a
c
n
s
v
y
population


class
e
p
4000
3500
3000
2500
2000
1500
1000
500
0
count
d
g
l
m
p
u
w
habitat

2.4 EDA conclusions

Looking at the above visualisations of the dataset, it’s already quite easy to see patterns among the mushrooms.

The bar charts were extremely useful to draw conclusions from since it’s very easy to see disparities between the two categories at a glance. For example, when it comes to a mushroom colony’s population, abundant and numerous mushrooms are all safe to eat, whereas if you came across a colony with only several in it, it’s probably not too safe.

Other standouts for edibility were either a lack of an odour or an almond smell, gilled mushrooms, a brown or white colour and if it was found in a meadow.

It’s very easy to see that there are a lot of mushroom properties that only show up in edible mushrooms. When judging if our models categorise the mushrooms correctly, we can refer to the fully edible or fully poisonous properties to see if the results are accurate.

There are also many variables that are rarer than others. While some properties are in the thousands, others are in double or even single digits. Hopefully the low weighting of the smaller values doesn’t affect the results in any way.

3 Experimental Design

3.1 Identification of your chosen supervised learning algorithm(s)

In order to classify the mushrooms into either edible or poisonous, I’ve chosen to use logistic regression. Logistic regression is a statistics-based model commonly used for classification or prediction. It analyses the probability of a given question being true or false based on certain variables. In this specific case, it will look through each variable of the mushroom, such as its colour, stalk, and gills to judge its edibility based on the training data given to it.

Logistic regression is a suitable choice for classifying these mushrooms since there’s only two end states for the mushrooms to be put into.

3.2 Identification of appropriate evaluation techniques

Alongside my logistic regression, we have Dan’s support vector machine, Kristine’s decision tree and Jack’s random forest.

A support vector machine is a versatile graph-based algorithm that tries to find a hyperplane, which is essentially a line of best fit, along the given data points, being the different mushrooms.

A decision tree works somewhat like a flowchart and goes down

a series of choices to eventually classify a given question. It works by creating a number of branching paths based on the variables fed to it.

The random forest algorithm functions a lot like the decision tree algorithm given that it makes use of numerous decision trees to come to a conclusion. The random forest utilises its many decision trees by running them all on the same data point and then vote based on the result given. In our case, it takes a mushroom and runs several decision trees on it. When each one is done, it takes the most common arrived at answer, either edible or inedible.

In order to validate our algorithms, we mainly considered its false positive rates and the accuracy in the final classification report.

False positive rates were a big standout for us. If this were a real world problem that we were aiming to solve, a false positive could result in either a hospitalisation or even a death, while accuracy would be an overall sign that our algorithm was correct.

3.3 Data cleaning and Pre-processing transformations

Our dataset luckily did not come with any missing values since every count totalled 8124, however, 2480 mushrooms had an unknown stalk type. How this came about was not mentioned, whether due to the mishandling of the mushrooms from the data collectors or if they were just found that way. I wouldn’t consider this missing data, although a stalk is a good way of analysing a mushroom’s edibility. These mushrooms can still work well for training the algorithm since they were all still able to be categorised into being edible or poisonous. It would be interesting to see if the results would be affected if we removed every mushroom with an unknown stalk, however.

There was also one redundant column. Each mushroom in the dataset had a universal veil and so we opted to remove it since it had no useful effect on the algorithm and could potentially lead to false positives, which is especially dangerous if someone decided to use this algorithm to eat an unidentified wild mushroom.

```
dfmushroom.drop(columns='veil-type', inplace=True)
```

This line of code will drop the veil-type variable from the table.

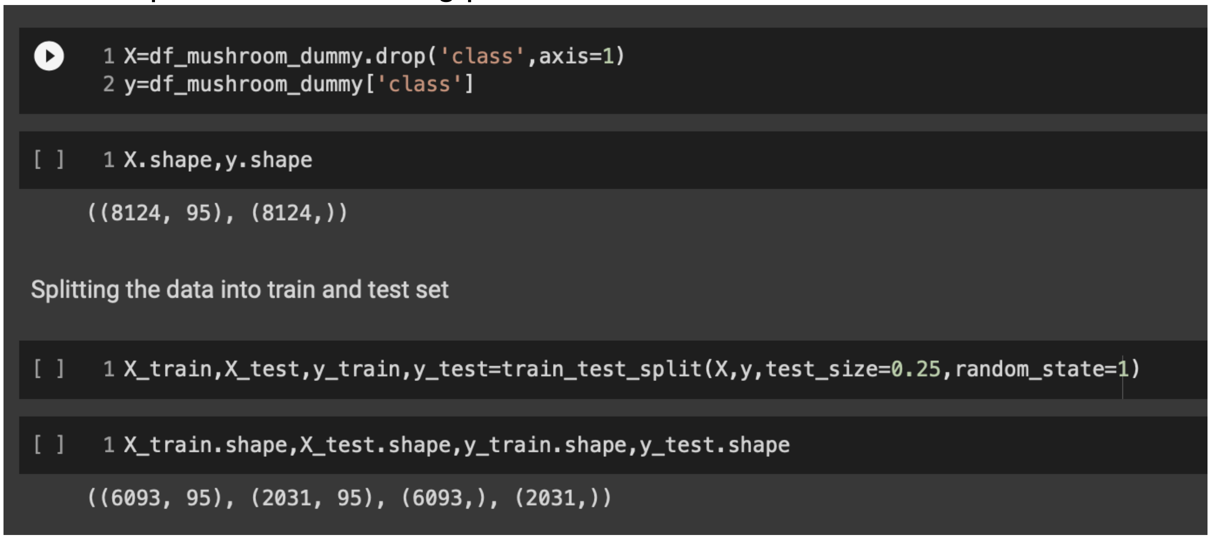
3.4 Limitations and Options

My chosen model, logistic regression, will never tell you which of the mushrooms given are safe to eat, only the probability that one will be. Unless a mushroom is for sale or confirmed by a certified mycologist, it shouldn’t be eaten blindly.

As mentioned previously in section 2.1, this will never be a definitive way to tell if you’re safe to just eat a mushroom, there will always be a risk involved with wild mushrooms with no one sign being a safe indicator. When 22 different characteristics are given, the safety of course will increase, but only up to a point.

4 Predictive Modelling / Model Development

4.1 The predictive modelling process



```
X=df_mushroom_dummy.drop('class',axis=1)
y=df_mushroom_dummy['class']
```

```
[ ] X.shape,y.shape
```

```
((8124, 95), (8124,))
```

Splitting the data into train and test set

```
[ ] X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=1)
```

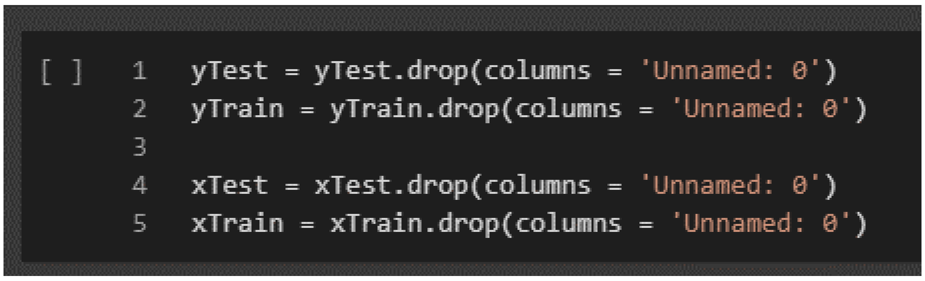
```
[ ] X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((6093, 95), (2031, 95), (6093,), (2031,))
```

75% of the dataset was used to train the model. As mentioned before in 3.3, we opted to drop the universal veil category due to redundancy and to avoid false positives leaving us with 22 different columns.

We chose to randomise the list rather than just splitting it. Although it seemed to already be random, we wanted to ensure we got a good spread of edible and inedible mushrooms with varying traits in both files. We also opted to separate the mushroom's class from the rest of the variables, leaving us with 21 different rows in X and 1 in Y.

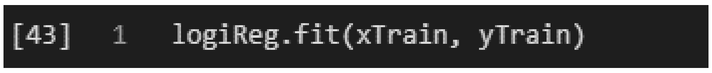
A final change we decided to make was to change the class's values from e and p to 0 and 1 to further emphasise that our algorithm was going to produce a black or white yes or no answer.



```
[ ] yTest = yTest.drop(columns = 'Unnamed: 0')
    yTrain = yTrain.drop(columns = 'Unnamed: 0')

    xTest = xTest.drop(columns = 'Unnamed: 0')
    xTrain = xTrain.drop(columns = 'Unnamed: 0')
```

We also had to drop the first column of the testing and training tables due to the way we exported them.

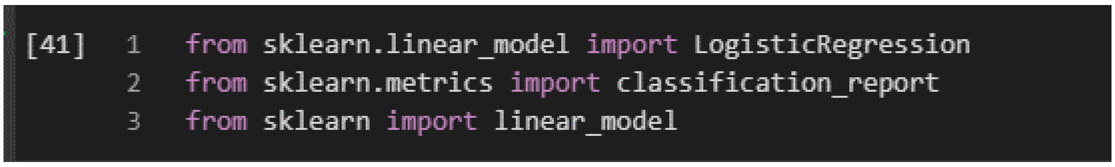
Finally, we can run .fit on the logistic regression algorithm to train it!



```
[43] logiReg.fit(xTrain, yTrain)
```

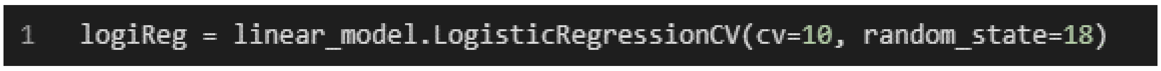
4.2 Evaluation results on “seen” data

Several libraries had to be imported before anything else.



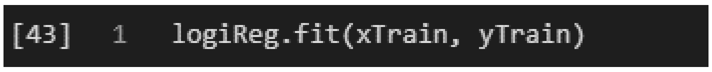
```
[41]  from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import classification_report
      from sklearn import linear_model
```

We then create our logistic regression model by calling sklearn’s LogisticRegressionCV. It’s possible to just run LogisticRegression, but the addition of CV allows it to run k-fold cross validation which separates the data into different groups to help avoid bias and veers away from any inaccurate predictions.



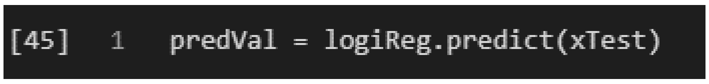
```
logiReg = linear_model.LogisticRegressionCV(cv=10, random_state=18)
```

.fit trains the algorithm on the training data before showing it the testing tables.



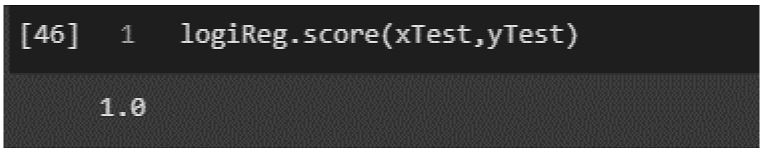
```
[43]  logiReg.fit(xTrain, yTrain)
```

.predict is used in the final report, it works somewhat like .fit but based on testing data instead,



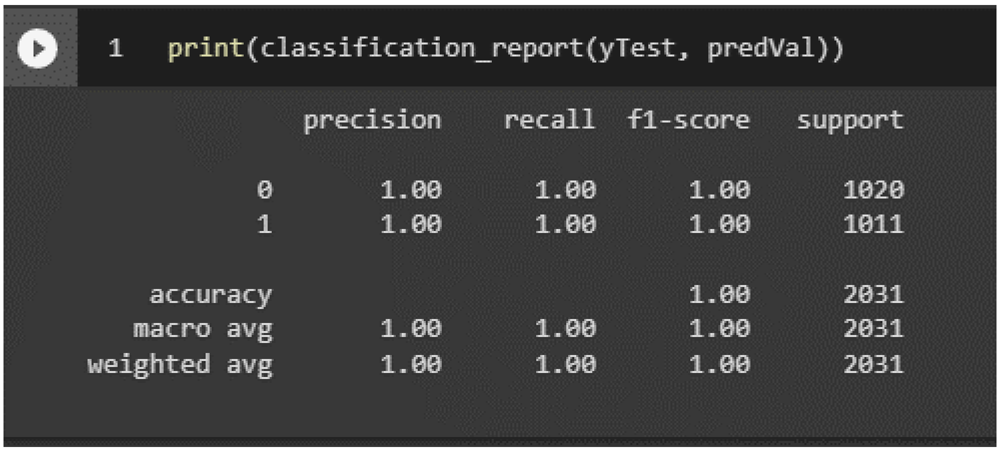
```
[45]  predVal = logiReg.predict(xTest)
```

.score creates a f1 score. A f1 score ranges from 0 to 1, with 1 being the most accurate and 0 being not accurate at all.



```
[46]  logiReg.score(xTest,yTest)

1.0
```

We used classification reports to compare our models. The precision represents its ability to avoid false positives, recall is how well it can find all the positives, f1 score is a weighted mean of the precision and recall. The macro average is the arithmetic mean between the f1 scores, while the weighted average adds them with a weight based on how many there are.

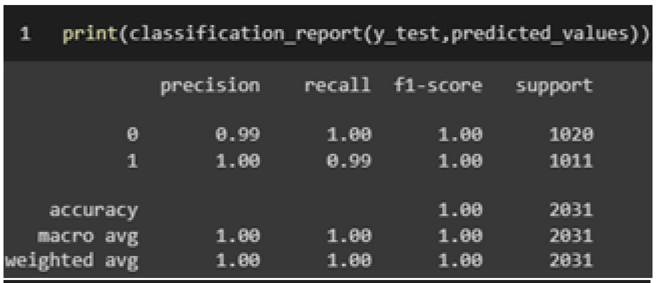


```
print(classification_report(yTest, predVal))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1020
           1       1.00      1.00      1.00      1011

    accuracy                           1.00      2031
   macro avg       1.00      1.00      1.00      2031
weighted avg       1.00      1.00      1.00      2031
```

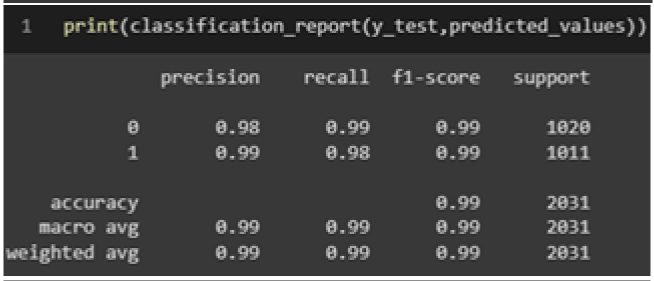
5 Evaluation and further modelling improvements

5.1 Initial evaluation comparison

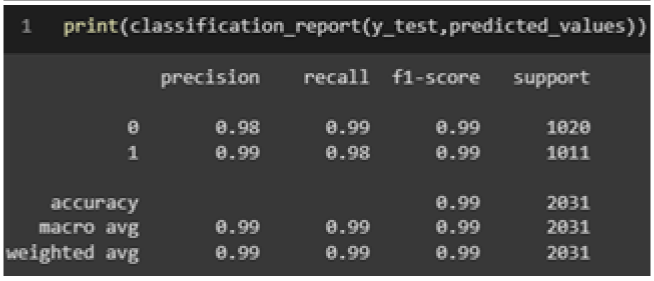
Dan ran three different SVM tests, the first being a linear kernel, the second being radial and the third being sigmoid. Although very slight, the radial and sigmoid methods had a slight chance of error. While this would normally be acceptable due to it only being off by 1% or 2%, this is dealing with poisonous foods, so only the linear kernel can be acceptable in this scenario.



```
print(classification_report(y_test,predicted_values))

              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1020
           1       1.00      0.99      1.00      1011

    accuracy                           1.00      2031
   macro avg       1.00      1.00      1.00      2031
weighted avg       1.00      1.00      1.00      2031
```

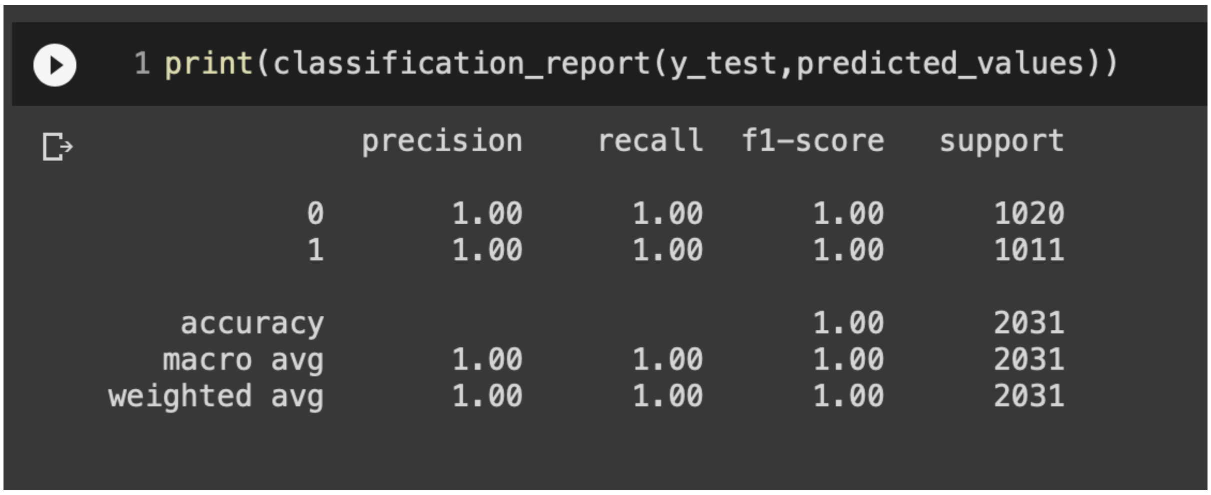


```
print(classification_report(y_test,predicted_values))

              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1020
           1       0.99      0.98      0.99      1011

    accuracy                           0.99      2031
   macro avg       0.99      0.99      0.99      2031
weighted avg       0.99      0.99      0.99      2031
```



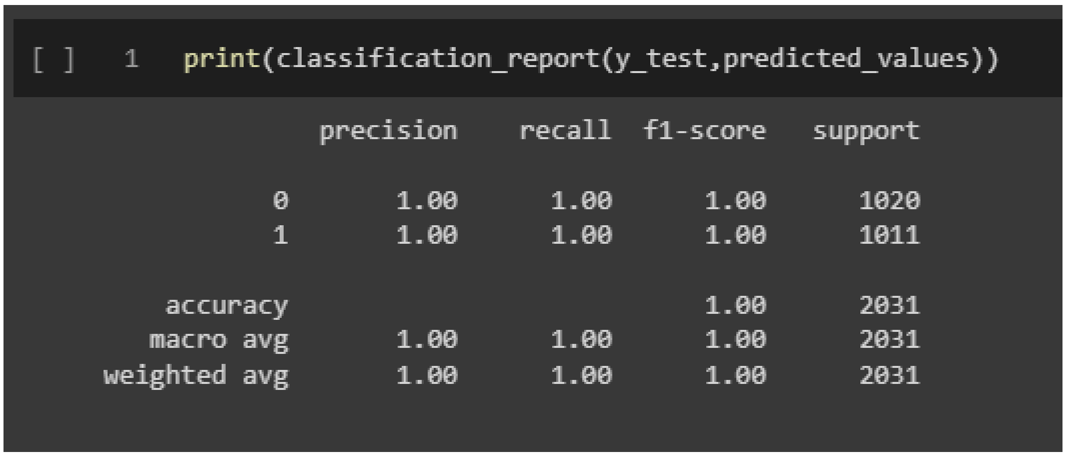
```
print(classification_report(y_test,predicted_values))

              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1020
           1       0.99      0.98      0.99      1011

    accuracy                           0.99      2031
   macro avg       0.99      0.99      0.99      2031
weighted avg       0.99      0.99      0.99      2031
```

Kris’s decision tree ended up with a 100% accuracy as well.



```
print(classification_report(y_test,predicted_values))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1020
           1       1.00      1.00      1.00      1011

    accuracy                           1.00      2031
   macro avg       1.00      1.00      1.00      2031
weighted avg       1.00      1.00      1.00      2031
```

Jack’s random forest also came out with 100% accuracy, which isn’t too surprising considering that a random forest algorithm is an ensemble of decision trees.



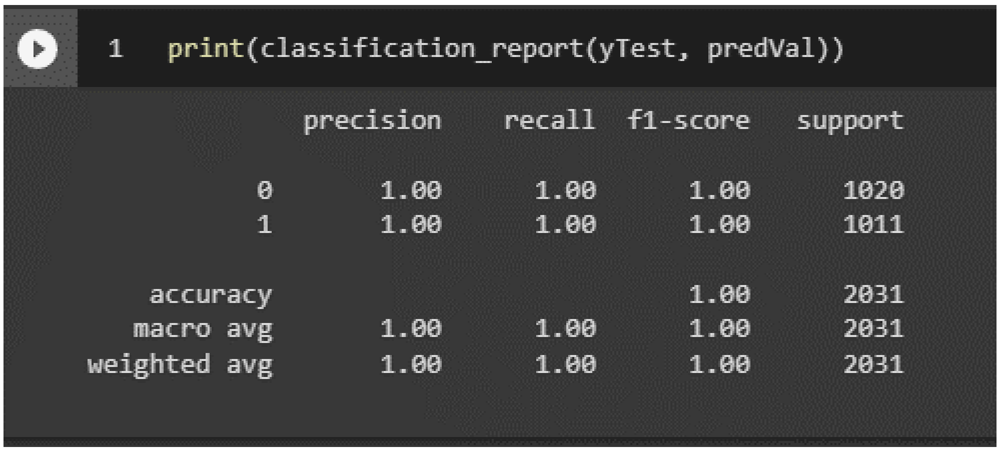
```
print(classification_report(y_test,predicted_values))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1020
           1       1.00      1.00      1.00      1011

    accuracy                           1.00      2031
   macro avg       1.00      1.00      1.00      2031
weighted avg       1.00      1.00      1.00      2031
```

5.2 Further modelling improvements attempted

Aside from two of Dan’s models, we all ended up with 100% accuracy on our model. There were small hiccups in our path, like how our split tables had an erroneous first column which we had to drop.

As I had mentioned before, just a bit over 2000 mushrooms had an unknown stem type. It might have been worth creating another testing table without the stems labelled with a ? to see if it had any impact on the results.

5.2 Final Evaluation results



```
print(classification_report(yTest, predVal))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1020
           1       1.00      1.00      1.00      1011

    accuracy                           1.00      2031
   macro avg       1.00      1.00      1.00      2031
weighted avg       1.00      1.00      1.00      2031
```

My own final results are the same as the others.

I can’t help but feel a little concerned that all of us ended up with 100% accuracy for our models. I do wonder if something went wrong in the data processing to give us such accurate results, however, no errors were thrown out and Dan did end up getting some erroneous results on his other attempts, meaning there’s a chance that nothing could have gone wrong either.

6 Conclusion

6.1 Summary of results

In terms of mushroom classification, linear regression, decision trees and random forests are all perfectly suited to sorting them due to our 100% accuracy. When it comes to support vector models, a linear kernel is best suited to also throwing out 100% accuracy, while both the radial and sigmoid SVMS had a small error, making it unsuitable due to the nature of mushrooms being potentially toxic.

6.2 Suggested further improvements to the model development process

While my model was accurate, there were a few more things I feel like I could have done more. As mentioned below, I wasn't too happy with the results of the EDA phase and I feel as though it could have been expanded on. Part of me wonders if the model really is accurate, I'm rather curious to see how the model with three outputs, edible, poisonous and unsure, would turn out.

6.3 Reflection on Research Team

It was really nice having a group to work with for this, they helped a lot especially when I was confused about certain things. We relegated the earlier parts of work to different team members, for example, while I found the data set, another looked for usable machine learning models to use, later on, another sorted out the dataset splitting. I feel like we worked really well as a team and were able to fill each other's gaps in understanding.

6.4 Reflection on Individual Learning

AI was something that I used to feel extremely intimidated by, although at the same time extremely fascinated in! It was nice to see that it was a lot more approachable than it seemed in this module. On a more self-indulgent level, I find mushrooms really cool and it was a nice change of pace to work on something that I enjoyed. In terms of the actual process, I feel like the EDA phase was fairly lackluster. I would have liked to try a few more graphs out to help visualise the different aspects, such as pie charts, for example.

7 References

UCI Machine Learning, 2016, Mushroom Classification.
[Available at: https://www.kaggle.com/datasets/uciml/mushroom-classification]
[Accessed on: 25/11/2022]

Sushree Sangeeta Jena, 2022, Model_Building_Mushrooms_dataset.
[Available at: https://www.kaggle.com/code/sushreesangeetajena/model-building-mushrooms-dataset/data]
[Accessed on: 25/11/2022]

Ahmed Gado, 2022, Mushroom Class: with 100% accuracy.
[Available at: https://www.kaggle.com/code/ahmedgadoo/mushroom-classes-with-100-accuracy]
[Accessed on: 25/11/2022]

IBM, Year Unknown, Logistic Regression
[Available at: https://www.ibm.com/uk-en/topics/logistic-regression/]
[Accessed on: 05/12/2022]

Eric Biggane, 2014, How to tell the difference between poisonous and edible mushrooms
[Available at: https://www.wildfooduk.com/articles/how-to-tell-the-difference-between-poisonous-and-edible-mushrooms/]
[Accessed on: 05/12/2022]

Rohith Gandhi, 2018, Introduction to Machine Learning Algorithms: Logistic Regression
[Available at: https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36]
[Accessed on: 10/12/2022]

Rohith Gandhi, 2018, Support Vector Machine — Introduction to Machine Learning Algorithms
[Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47]
[Accessed on: 13/01/2023]

Tony Yiu, 2019, Understanding Random Forest
[Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2]
[Accessed on: 13/01/2023]

Kopal Jain, 2021, How to improve logistic regression
[Available at: https://medium.com/analytics-vidhya/how-to-improve-logistic-regression-b956e72f4492]
[Accessed on: 14/01/2023]

Mohita Narang, 2023, K-fold Cross-Validation
[Available at: https://www.naukri.com/learning/articles/k-fold-cross-validation/]
[Accessed on: 14/01/2023]

Muthukrishnan, 2018, Understanding the Classification report through sklearn
[Available at: https://muthu.co/understanding-the-classification-report-in-sklearn/]
[Accessed on: 15/01/2023]