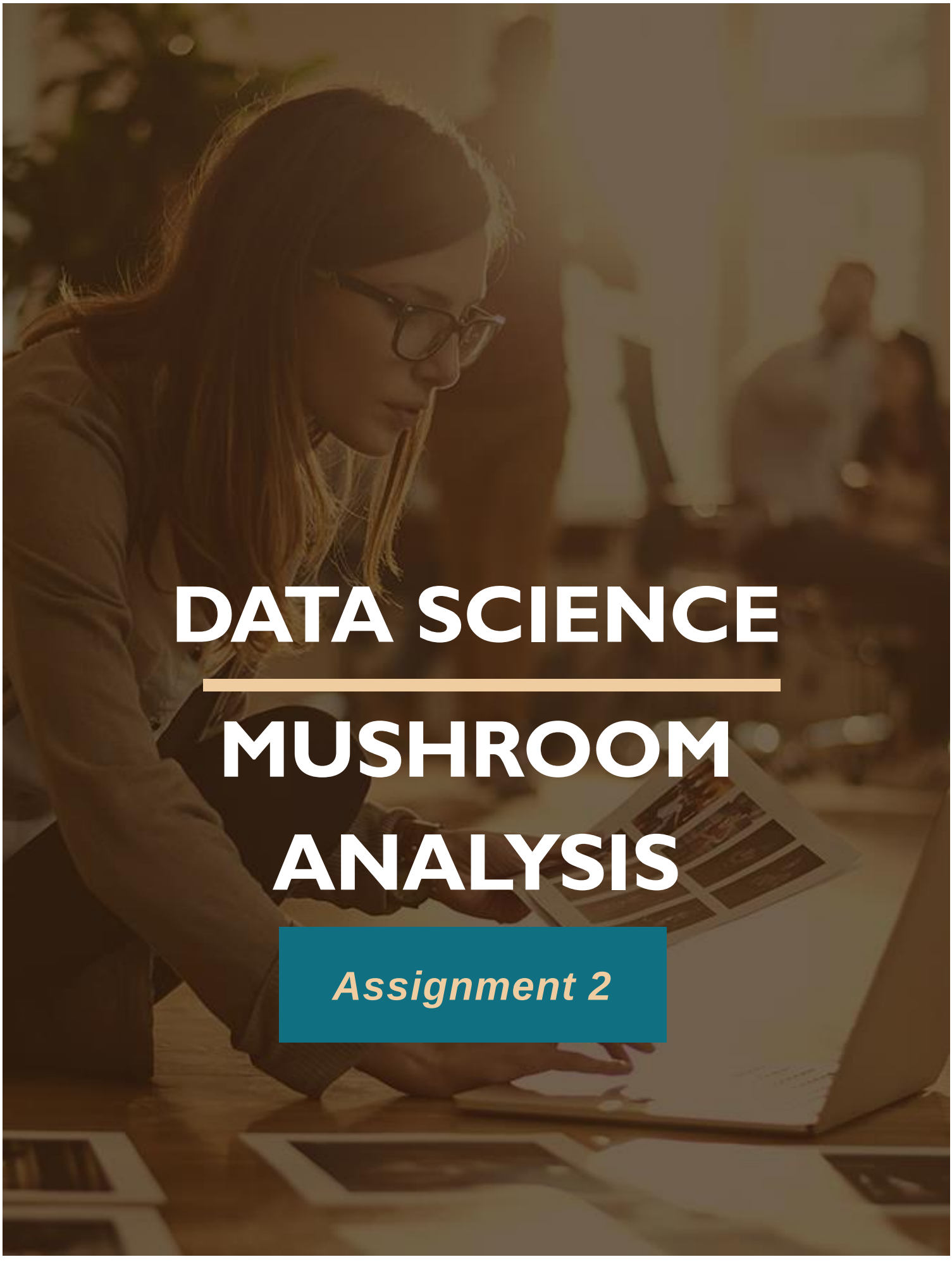


DATA SCIENCE

MUSHROOM ANALYSIS

Assignment 2

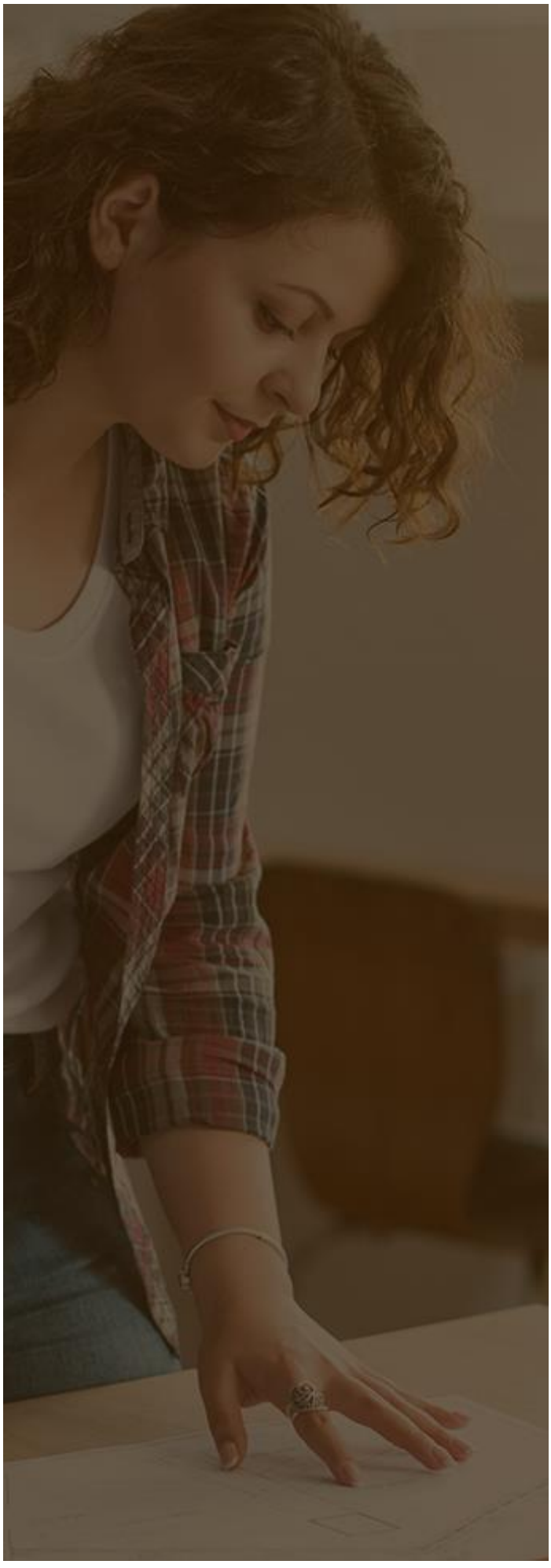


TABLE OF CONTENTS

INTRODUCTION

This data science project is based on the samples of 23 species of gilled mushrooms provided by the Agaricus and Lepiota Family Mushroom taken from The Audubon Society Field Guide to North American Mushrooms (1981). The key goals of this project are as below

- To select the most suitable prediction models to analyze and produce the outcome
- To find the best features to indicate the presence of poisonous mushrooms.
- To find out which mushroom is eatable

This project uses **python** to process, analyse, and create the prediction model.

The key libraries used for the project are as below

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
import graphviz

#Prediction models library
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.metrics import classification_report, confusion_matrix, precision_recall_curve, auc, roc_curve
```

DATA PREPARATION

In the first stage, the mushroom.csv file was imported and read

```
df = pd.read_csv("mushrooms.csv")
df.head()
```

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

5 rows × 23 columns

From examining the csv file, there are **no** missing elements and therefore do not need to be treated.

To find the attributes of the mushroom, the following code was run and results displayed below

```
df.info()
df.describe()
```

```
<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)
```

```
df.shape
```

```
(8124, 23)
```

Based on the results, there are **23** attributes from a cumulation of **8124** mushroom types. Next, to find out which mushroom is poisonous and eatable, the following code is run and displayed below. The **column** class attribute was used to filter out which mushroom is unique and abbreviated.

- e – eatable
- P - poisonous

```
df['class'].unique()
```

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

```
df['class'].value_counts()
```

```
e    4208
p    3916
Name: class, dtype: int64
```

There is a total of **4208** eatable and **3916** poisonous mushrooms. This attribute will be used later for the prediction model to identify how many are actually eatable or poisonous

This step begins the data pre-processing which consists of (removing outliers and conversion of categorical to numerical data)

- Outliers – These data are removed because outliers are unusual elements and cause statistical data analysis to be distorted. Common assumptions may be violated also. In this dataset, the **veil-type** is an outlier because most of it is 0.

- Categorical to Numerical – It is important to convert these values to numerals because categorical data may distort results by prediction models.

After the conversion and pre-processing, the results are displayed below. Other visualization processes include violin, correlation, and factor plot.

In [8]:
```
labelencoder=LabelEncoder()
for column in df.columns:
    df[column] = labelencoder.fit_transform(df[column])
```

In [9]:
```
df.head()
```

Out[9]:

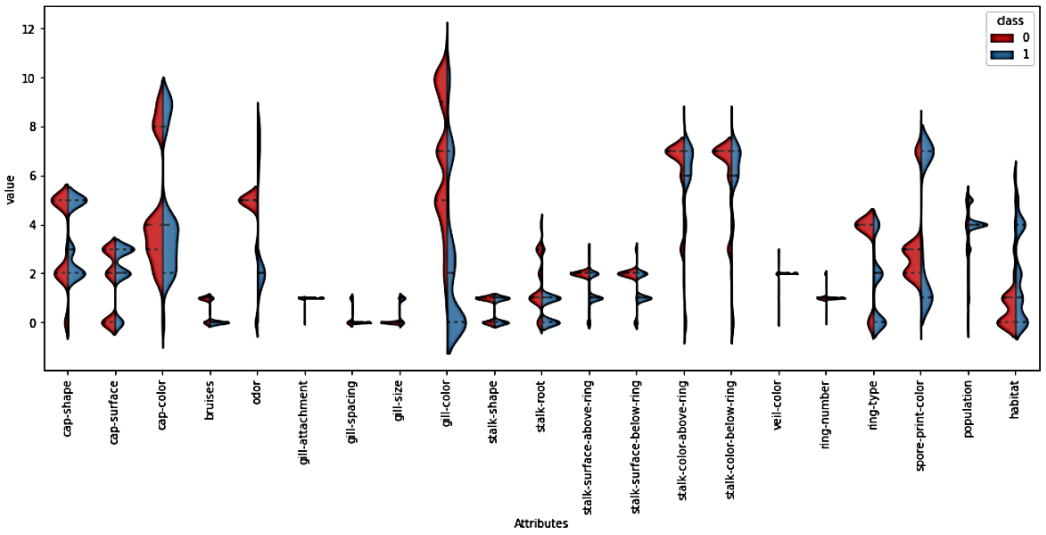
	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	population
0	1	5	2	4	1	6	1	0	1	4	...	2	7	7	0	2	1	4	2	3
1	0	5	2	9	1	0	1	0	0	4	...	2	7	7	0	2	1	4	3	2
2	0	0	2	8	1	3	1	0	0	5	...	2	7	7	0	2	1	4	3	2
3	1	5	3	8	1	6	1	0	1	5	...	2	7	7	0	2	1	4	2	3
4	0	5	2	3	0	5	1	1	0	4	...	2	7	7	0	2	1	0	3	0

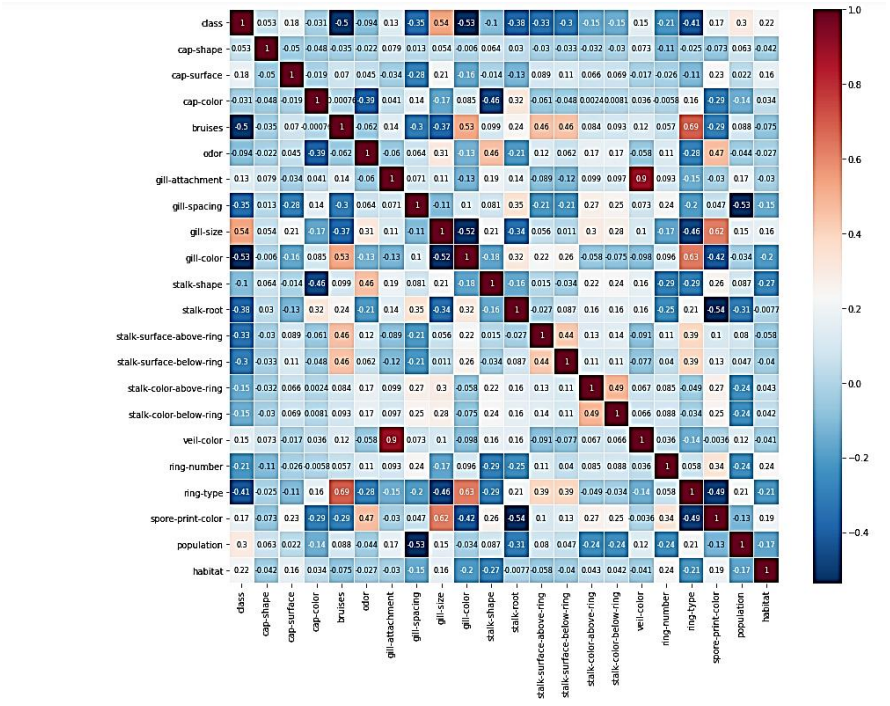
5 rows × 23 columns

In [10]:
```
df['veil-type'].unique() # 0
df = df.drop(['veil-type'],axis=1)
df.columns
```

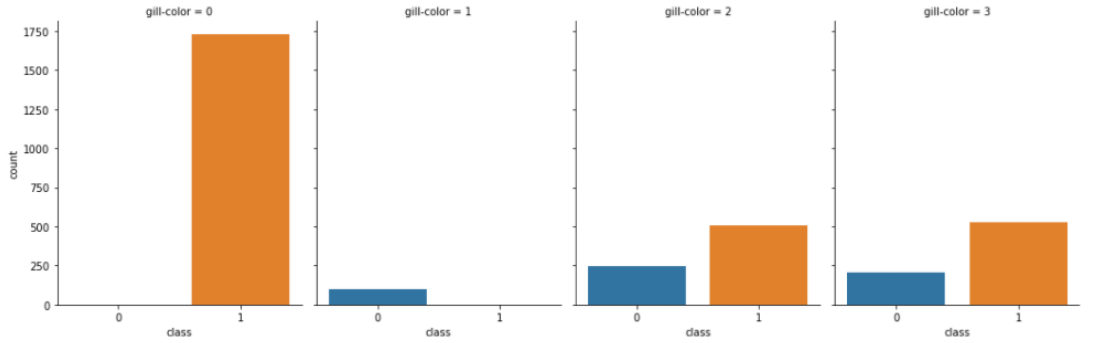
Out[10]:
```
Index(['class', 'cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor',
       'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color',
       'stalk-shape', 'stalk-root', 'stalk-surface-above-ring',
       'stalk-surface-below-ring', 'stalk-color-above-ring',
       'stalk-color-below-ring', 'veil-color', 'ring-number', 'ring-type',
       'spore-print-color', 'population', 'habitat'],
      dtype='object')
```

In [13]:
```
df_div = pd.melt(df, "class", var_name="Attributes")
fig, ax = plt.subplots(figsize=(16,6))
p = sns.violinplot(ax = ax, x='Attributes', y='value', hue='class', split = True, data=df_div, inner = 'quartile', palette = 'Set
df_no_class = df.drop(["class"],axis = 1)
p.set_xticklabels(rotation = 90, labels = list(df_no_class.columns))
plt.show()
```





```
In [15]: new_var = df[['class', 'gill-color']]
         new_var = new_var[new_var['gill-color']<=3.5]
         sns.catplot('class', col='gill-color', data=new_var, kind='count', height=4.5, aspect=.8, col_wrap=4)
         plt.show()
```



From the above visualizations, the veil-type outlier is dropped, further outlier analysis through violin plot, and data correlation, factor plot visualization. Based on the visualizations, it can be observed that gill-color has a correlation of -0.53 and an outlier. It shows that there is the **least** correlation especially between gill color 0 & class 1 mushroom.

PREDICTION MODEL

Next, to prepare the dataset for the model, the steps are displayed below

```
In [16]: X = df.drop('class',axis=1)
         y = df['class']

In [17]: X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33, test_size=0.1)

In [33]: y_train
         test_training_datasets = {"X_train":X_train, "y_train":y_train, 'X_test':X_test, "y_test":y_test, }

In [34]: for i,j in test_training_datasets.items():
           print(i,j.shape)

         X_train (7311, 21)
         y_train (7311,)
         X_test (813, 21)
         y_test (813,)
```

The data is split into training and test data set. The training data is **90%** and test is **10%** of the data. Randomization of data in **train_test_split** function is achieved through the **random_state = 33.** The 33 is used because each pair of dataset consists of **1/3** of the entire data set.

The **logistic regression** model is used, because based on the visualization, the data is very linear in nature. In addition, the values are within 1- 10 which is suitable for regressions. In an article by (Hseltman, 2018, Experimental Design & Analysis), logistic regression are used commonly because of the flexibility in testing the relationships between quantitative and categorical variables which produces a binary/single categorical outcome on output

The library used for the regression model is from **sklearn**.

```
In [21]: from sklearn.linear_model import LogisticRegression
         lr = LogisticRegression(solver='lbfgs',max_iter=500)
         lr.fit(X_train,y_train)

Out[21]: LogisticRegression(max_iter=500)
```

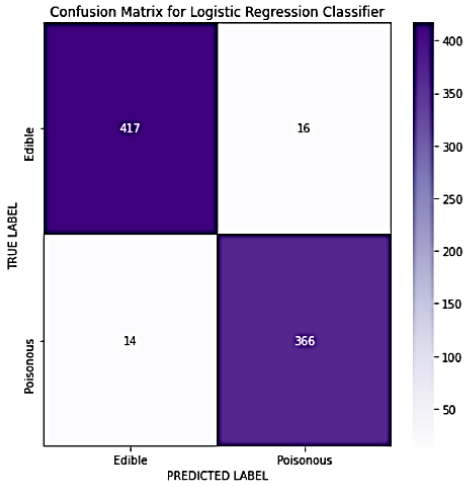
In [22]:
```
y_pred_lr = lr.predict(X_test)
print("Logistic Regression Classifier Report: \n", classification_report(y_test, y_pred_lr))
```

```
Logistic Regression Classifier Report:
               precision    recall  f1-score   support

           0       0.97      0.96      0.97       433
           1       0.96      0.96      0.96       380

    accuracy                           0.96       813
   macro avg       0.96      0.96      0.96       813
weighted avg       0.96      0.96      0.96       813
```

In [23]:
```
cm = confusion_matrix(y_test, y_pred_lr)
x_axis_labels = ["Edible", "Poisonous"]
y_axis_labels = ["Edible", "Poisonous"]
f, ax = plt.subplots(figsize =(7,7))
sns.heatmap(cm, annot = True, linewidths=0.2, linecolor="black", fmt = ".0f", ax=ax, cmap="Purples", xticklabels=x_axis_labels, y
plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for Logistic Regression Classifier')
plt.show()
```



In [24]:
```
print(f"Accuracy {round(lr.score(X_test,y_test),100*2)*100}%")
```

```
Accuracy 96.30996309963098%
```

```
pred = lr.predict(X_test)
print(pred[3:20])
print(y_test[3:20].values)
```

```
[0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0]
[0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0]
```

RESULTS AND CONCLUSION

We can conclude that the accuracy of the prediction model on all repeat sets is 96.3099% +-. To further understand the confusion matrix, the explanations are as below:

Edible – 417 (True Positive)

Edible but poisonous – 16 (False Positive)

Poisonous but edible – 14 (False Negative)

Poisonous – 366 (True Negative)

REFERENCES

Stat.cmu.edu. 2021. [online] Available at: <http://www.stat.cmu.edu/~hseltman/309/Book/Book.pdf> [Accessed 22 May 2021].

Medium. 2021. *Understanding Confusion Matrix*. [online] Available at: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62> [Accessed 22 May 2021].

021. [online] Available at: <https://statisticsbyjim.com/basics/remove-outliers/> [Accessed 22 May 2021].

Medium. 2021. *Mushroom Classification Using Different Classifiers*. [online] Available at: <https://medium.com/analytics-vidhya/mushroom-classification-using-different-classifiers-aa338c1cd0ff> [Accessed 22 May 2021].

Kaggle.com. 2021. *Mushroom Classification*. [online] Available at: <https://www.kaggle.com/uciml/mushroom-classification/code> [Accessed 22 May 2021].

Huq, R., 2017. Training a machine to determine whether a mushroom is edible. *In Machines We Trust*. Available at: https://inmachineswetrust.com/posts/mushroom-classification/ [Accessed May 22, 2021].