November 20, 2023

Mushroom Edibility Classifier

C964: Computer Science Capstone

Part A: Letter of Transmittal

11/20/2023

Sebastian Schmitt

Mycological Research Association (MRA)

123 Park Avenue, New York, NY

Dear Sebastian,

As you and the rest of us here at the MRA know, the organization cannot continue its course with the limited income streams it currently possesses since we are sustaining an outrageous 20% decrease in revenue annually. It is clear now more than ever that we are in desperate need of more ways to generate and increase revenue as well as ways to decrease wasted time and effort. However, it seems to me that it would be ideal to do this in such a way as to leverage our organization's strengths.

This brings me to a proposal for a new mushroom edibility classification system that utilizes our knowledge of mycological specimens. The core purpose of this application would be to display a message to users explaining whether a particular mushroom is "edible" or "poisonous" based on user input of a mushroom's mycological characteristics.

The application would be an excellent way to provide revenue for the organization since we would be able to market it for consumer use via subscriptions or software sales. Also, we could use it internally within the MRA to decrease the amount of time spent identifying the edibility of fungi by giving our researchers an initial prediction that they can include in their mushroom classification process.

It will take approximately a month to develop a working proof of concept for this app, and it will cost less than 9k to produce. This is because the only cost that would be required is the time and effort of a software developer, and we already have a lot of the infrastructure necessary to handle such an undertaking. Also, the data that will be used will either consist of the MRA's existing data, open-source data, or a combination of both. Fortunately, there are no ethical complications with either of these options.

As Senior Assistant Research Manager of the MRA and a Computer Scientist, I have worked on several projects similar to the one I am proposing and possess the skill and educational background to confirm that such a solution is feasible and worthy of consideration.

Please reach out to me if you have any questions.

Sincerely,

Senior Assistant Research Manager

Part B: Executive Summary

Project Summary

The Mycological Research Association (MRA) is an organization dedicated to the study of mushrooms. However, their revenue is decreasing at a rate of 20% annually, and they need to decrease this rate of decline to continue their research. They are looking for new income streams and would like to implement ones that utilize their expertise.

The purpose of this proposal is to put forth a proof of concept for a mushroom edibility classifier that will aid the organization and individuals in the classification of mushrooms by giving edibility predictions based on specific mycological characteristics. To do this, we will create a thorough Jupyter Notebook and a user guide. This notebook will include code for the acquisition, cleaning, processing, exploration, and modeling of mushroom classification data; a performance report of the application's machine-learning model; and a simple user interface for predicting the edibility of a particular mushroom.

We expect this new system to increase revenue for the organization by 20% since they could easily find ways to market it for consumer use via subscriptions or software sales. Also, we can see this system being used internally to increase the accuracy of mushroom edibility classification and decrease the time it takes.

Data Summary

The dataset will be downloaded directly from Kaggle.com as a CSV file, uploaded into a Jupyter Notebook, and contain mushroom classification data consisting of 23 different columns with each one representing a specific mycological characteristic. The data will be categorical, and each cell will contain a letter that represents a type of mycological characteristic represented in the column. For example, the characteristic, “class,” would be represented with either “e” for edible or “p” for poisonous.

The data will then be read from the CSV file and placed into a Pandas DataFrame labeled "data." This will hold the original data for the entirety of the project. The data will then be copied from this DataFrame, and this copy will be modified to make the data usable for exploration and modeling by converting it into numerical data. For example, "e" would be converted to 0, and "p" would be converted to 1.

Once the data has been converted into numerical data, we can then process the data for modeling. This will involve splitting the data up into two sets of data: one for training and one for testing. Furthermore, these two sets of data will have their "class" and "veil-type" columns removed with their "class" columns being placed into separate datasets to be used in the training and testing processes.

Finally, a single row of data will be created for a new mushroom that a user will input mycological values by using Python. This row will be concatenated to a dataset so that its values can be properly converted into numerical data. We will then be able to take this new dataset, use our model to predict each row's edibility, and then use the prediction of the last row (which we created) to print a message describing our inputted mushroom's edibility.

Because each row of "data" describes a particular mushroom and has a column indicating the fungus' edibility, this data is perfect for the project. The goal of this project is to classify whether a mushroom is edible or poisonous, and we can use all the columns that contain information on the mushroom's characteristics to predict the column with the mushroom's edibility classification. Also, there are no legal or ethical concerns with using this data, and it is freely accessible to the public since Kaggle.com is a public website.

Implementation

For this project, we will use the waterfall development methodology. This method will work well for the project because the requirements for the project are well-defined, and they are unlikely to change throughout its course.

The project will progress through the following standard waterfall phases:

1. Requirements:
    - Meeting with all stakeholders.
    - Gathering and defining requirements that need to be met.

2. Design:
    - Determining the programming language, external libraries, and software tools needed.
    - Deciding the application format for the system i.e., launched via IDE.
    - Selecting a machine-learning model to use that fulfills the requirements.
    - Deciding the data to be gathered.

3. Implementation:
    - Establishing a development environment.
    - Importing external Libraries and Data.
    - Performing Data Cleaning to remove any anomalies that may exist in the data.
    - Preparing the data for exploration and visualization.
    - Performing Exploratory Analysis on the data.
    - Preparing the data for modeling.
    - Logistic Regression training.
    - Developing a thorough Performance Report of the model.
    - Developing a simple User Interface for predicting the edibility of a particular mushroom using the model.

4. Testing:
    - Unit testing.
    - Integration testing.
    - Acceptance testing.

5. Deployment:
    - Releasing the application for public use.

6. Maintenance:
    - Acquiring user feedback
    - Analyzing application performance for improving the application in future versions.

Timeline

Milestone or deliverable	Duration (hours or days)	Projected start date	Anticipated end date
Environment Setup	4	11/1/2023	11/1/2023
Data Collection	14	11/2/2023	11/4/2023
Data Preprocessing	8	11/6/2023	11/6/2023
Data Exploration and Visualization	24	11/7/2023	11/11/2023
Model Training	16	11/13/2023	11/14/2023
Model Evaluation	8	11/15/2023	11/16/2023
User Interface Development	16	11/17/2023	11/18/2023
Testing	16	11/20/2023	11/21/2023
Documentation	8	11/22/2023	11/22/2023
Deployment	12	11/23/2023	11/25/2023

Evaluation Plan

The machine learning model that will be used for this project will be evaluated for its accuracy by using the following methods:

- Accuracy scores that are decimal values between 0 and 1.
- Confusion matrices that visualize the number of false positives, false negatives, true positives, and true negatives that the model predicts.
- Classification reports outlining the accuracy, precision, macro averages, and weighted averages of the model.

Once the model has been determined to be adequately accurate, we will use it to predict the edibility of a new mushroom whose mycological values will be inputted by a user. To verify that the final prediction of this mushroom's edibility is accurate, we will check to see if the prediction we are giving is for the correct fungus and not another one in the dataset. This will be done manually by comparing datasets.

Once the application is completed, it will also be evaluated as a whole for ease of use, overall functionality, and visualization presentation. This will be done primarily during the Acceptance testing phase when we begin comparing what we have created to stakeholder requirements.

Resources and Costs

Resource	Description	Cost
Labor	Developer hours	$8,820
Workstations	Already provided by the organization	$0
Server	Already provided by the organization	$0
Network	Already provided by the organization	$0
Jupyter Notebook	Open-source IDE accessible via Anaconda	$0
Anaconda	A distribution package of data science software	$0
Python 3.11	Programming language	$0
External Libraries	Open-source Python libraries such as Pandas, NumPy, Seaborn, Sklearn, etc.	$0
Data	Dataset from Kaggle.com	$0
Deployment	Sharing the Jupyter Notebook file	$0
	Total	$8,820

Part D: Post-implementation Report

Solution Summary

The Mycological Research Association (MRA) needs more revenue to continue their research, and to generate this revenue they are looking for new income streams that utilize their expertise. To do this, we developed an application that was marketed to the public and monetized by selling subscriptions for its use, which increased organizational revenue by 25% annually. The application accepts user input, which consists of a set of different mycological features, and outputs a message stating whether the mushroom is poisonous or edible.

Data Summary

The dataset was downloaded directly from Kaggle.com as a CSV file, uploaded into a Jupyter Notebook, and contained mushroom classification data consisting of 23 different columns with each one representing a specific mycological characteristic. The data was categorical, and each cell contained a letter that represented a type of mycological characteristic represented in the column. For example, the characteristic, “class,” was represented with either “e” for edible or “p” for poisonous, as demonstrated by the image below.

```
data
```

	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	popu
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8119	e	k	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o	p	b	
8120	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	n	o	p	b	
8121	e	f	s	n	f	n	a	c	b	n	...	s	o	o	p	o	o	p	b	
8122	p	k	y	n	f	y	f	c	n	b	...	k	w	w	p	w	o	e	w	
8123	e	x	s	n	f	n	a	c	b	y	...	s	o	o	p	o	o	p	o	

8124 rows × 23 columns

The data was then read from the CSV file and placed into a Pandas DataFrame labeled "data." This held the data and maintained an unchanged version of it for the entirety of the project. The data was then copied from this DataFrame, and this copy was modified later to make the data usable for exploration and modeling by converting it into numerical data. For example, "e" was converted to 0, and "p" was converted to 1 (refer to the image below).

```
df2 = data.copy()
for i in df2.columns:
    df2[i] = df2[i].astype('category')
    df2[i] = df2[i].cat.codes
df2
```

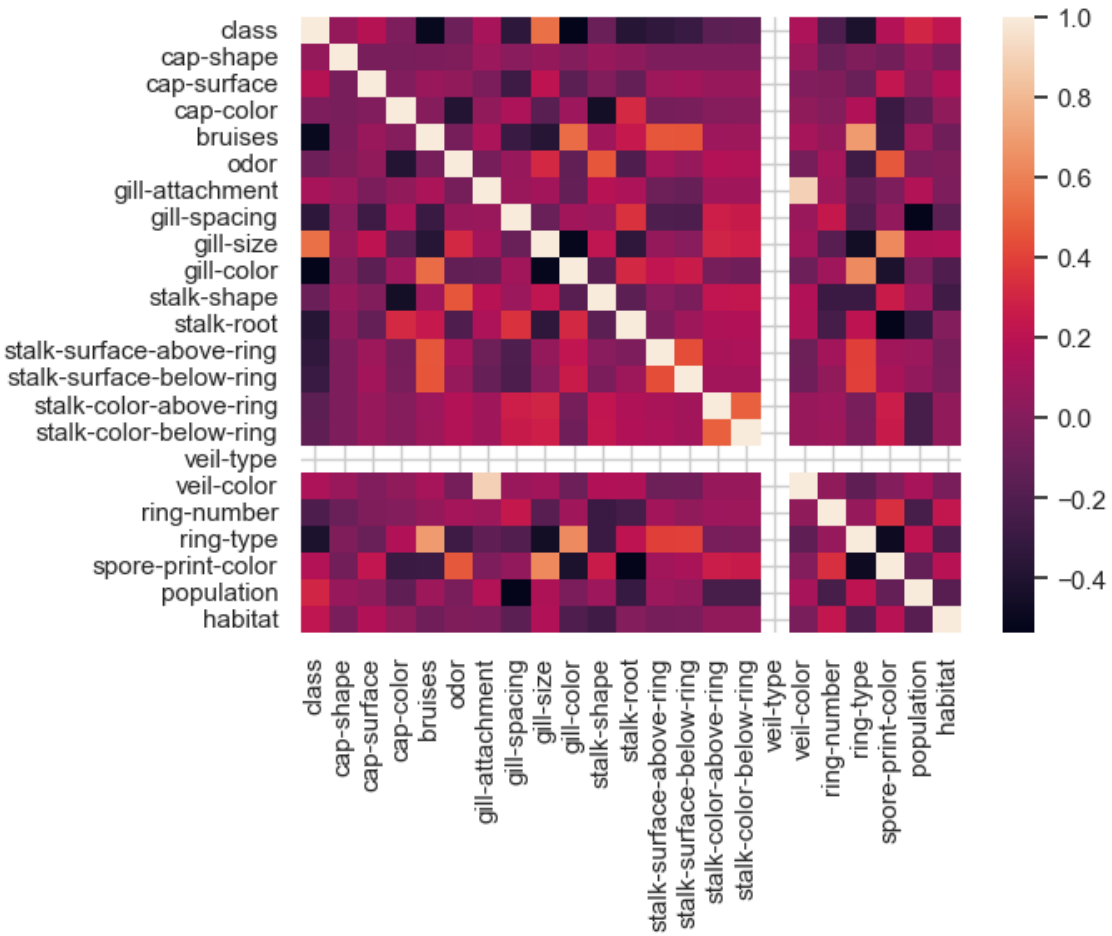
	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	popu
0	1	5	2	4	1	6	1	0	1	4	...	2	7	7	0	2	1	4	2	
1	0	5	2	9	1	0	1	0	0	4	...	2	7	7	0	2	1	4	3	
2	0	0	2	8	1	3	1	0	0	5	...	2	7	7	0	2	1	4	3	
3	1	5	3	8	1	6	1	0	1	5	...	2	7	7	0	2	1	4	2	
4	0	5	2	3	0	5	1	1	0	4	...	2	7	7	0	2	1	0	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
8119	0	3	2	4	0	5	0	0	0	11	...	2	5	5	0	1	1	4	0	
8120	0	5	2	4	0	5	0	0	0	11	...	2	5	5	0	0	1	4	0	
8121	0	2	2	4	0	5	0	0	0	5	...	2	5	5	0	1	1	4	0	
8122	1	3	3	4	0	8	1	0	1	0	...	1	7	7	0	2	1	0	7	
8123	0	5	2	4	0	5	0	0	0	11	...	2	5	5	0	1	1	4	4	

8124 rows × 23 columns

Once the data was converted into numerical data, we then processed the data for modeling. This involved splitting the data up into two sets. The first one, called "y," included only the "class" column, and the second, called "X," included all the columns except "class" and "veil-type."

```
y = df2['class']
X = df2.drop(columns = ['class', 'veil-type'])
```

"Class" was excluded from "X" because this was the column we were trying to predict, and "veil-type" was excluded because it didn't affect the output of the model (refer to the heatmap below).



We then used these two datasets to split the data into training and testing data by using Sklearn's "train_test_split" function. We passed "X" and "y" into this function, and it outputted four datasets called "X_train," "X_test," "y_train," and "y_test." "X_train" and "X_test" were subsets of "X" but were completely different from one another. The case was the same for "y_train" and "y_test" concerning "y."

It is also important to note that "X_train" and "y_train" had the same number of rows, and each row of X_train was directly related to the row of "y_train" that had the same index number. The idea was that "y_train" would contain the correct values that X_train was supposed to predict using its data. For example, the values of the first row of "X_train" would predict the first value of "y_train." This was also the same for "X_test" and "y_test."

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 42)
```

```
X_train
```

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	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
1224	2	2	4	0	5	1	1	0	7	1	...	0	2	7	7	2	1	0	2
142	4	0	4	0	5	1	0	1	4	0	...	2	2	7	7	2	1	4	3
4220	2	0	9	0	2	1	0	0	7	0	...	1	1	6	4	2	1	2	1
1490	2	0	3	0	5	1	1	0	7	1	...	0	2	7	7	2	1	0	2
6381	2	2	4	0	2	1	0	1	0	1	...	1	2	6	6	2	1	0	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5226	5	3	4	0	2	1	0	1	0	1	...	1	2	6	7	2	1	0	7
5390	3	3	2	1	5	1	0	0	10	0	...	2	2	7	2	2	2	0	7
860	2	3	4	1	3	1	0	0	10	0	...	2	3	7	7	2	1	4	3
7603	3	2	2	0	2	1	0	1	0	1	...	2	2	6	6	2	1	0	7
7270	3	0	3	0	5	1	1	0	2	0	...	2	1	7	7	2	2	4	7

5416 rows × 21 columns

```
y_train
```

```
1224    0
142     0
4220    1
1490    0
6381    1
       ..
5226    1
5390    0
860     0
7603    1
7270    0
Name: class, Length: 5416, dtype: int8
```

```
X_test
```

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	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
1971	2	0	4	0	5	1	1	0	3	1	...	2	0	7	7	2	1	0	3
6654	2	2	2	0	8	1	0	1	0	1	...	2	2	6	6	2	1	0	7
5606	5	3	4	0	2	1	0	1	0	1	...	1	2	7	6	2	1	0	7
3332	2	3	3	1	5	1	0	0	5	1	...	2	2	3	6	2	1	4	3
6988	2	2	2	0	7	1	0	1	0	1	...	2	2	6	6	2	1	0	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4816	2	0	9	0	2	1	0	0	3	0	...	1	1	4	0	2	1	2	1
2431	5	3	2	1	5	1	0	0	5	1	...	2	2	7	3	2	1	4	2
7871	0	2	4	0	5	0	0	0	11	0	...	2	2	5	5	1	1	4	0
564	0	2	8	1	0	1	0	0	10	0	...	2	2	7	7	2	1	4	3
2526	5	0	3	1	5	1	0	0	9	1	...	2	2	6	7	2	1	4	3

2708 rows × 21 columns

```
y_test
```

```
1971    0
6654    1
5606    1
3332    0
6988    1
       ..
4816    1
2431    0
7871    0
564     0
2526    0
Name: class, Length: 2708, dtype: int8
```

Finally, to predict the edibility of a user-defined fungus, a single row of data was created by inputting a particular mushroom's mycological values. This was done by merely creating 21 Jupyter Notebook cells that assign values to 21 variables using Python.

User Interface

Run the following cells and input the letter for the the mycological characteristic that best describes the fungus.

```
In [*]: cap_shape = str(input("Choose the cap shape from the following: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s: "))
```

```
Choose the cap shape from the following: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s:
```

```
In [*]: cap_surface = str(input("Choose the cap surface from the following: fibrous=f, grooves=g, scaly=y, smooth=s: "))

In [*]: cap_color = str(input("Choose the cap color from the following: brown=n, buff=b, cinnamon=c, gray=g,green=r, pink=p, purple=u

In [*]: bruises = str(input("Does it bruise? true=t, false=f: "))

In [*]: odor = str(input("Choose the odor from the following: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungen

In [*]: gill_attachment = str(input("Choose the gill attachment type from the following: attached=a, free=f: "))

In [*]: gill_spacing = str(input("Choose the gill spacing type from the following: close=c, crowded=w: "))

In [*]: gill_size = str(input("Choose the gill size from the following: broad=b, narrow=n: "))

In [*]: gill_color = str(input("Choose the gill color from the following: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, ora

In [*]: stalk_shape = str(input("Choose the stalk shape from the following: enlarging=e, tapering=t: "))

In [*]: stalk_root = str(input("Choose the stock root type from the following:  bulbous=b, club=c, equal=e, rooted=r, missing=?: "))

In [*]: stalk_surface_above_ring = str(input("Choose the kind of stalk surface that is above the ring from the following: fibrous=f,

In [*]: stalk_surface_below_ring = str(input("Choose the kind of stalk surface that is below the ring from the following: fibrous=f,

In [*]: stalk_color_above_ring = str(input("Choose the color of the stalk that is above the ring from the following: brown=n,buff=b,c

In [*]: stalk_color_below_ring = str(input("Choose the color of the stalk that is below the ring from the following: brown=n, buff=b,

In [*]: veil_color = str(input("Choose the veil color from the following: brown=n, orange=o, white=w, yellow=y: "))

In [*]: ring_number = str(input("Choose the number of rings from the following: none=n, one=o, two=t: "))

In [*]: ring_type = str(input("Choose the ring type from the following: evanescent=e, flaring=f, large=l, none=n, pendant=p: "))

In [*]: spore_print_color = str(input("Choose the spore print color from the following: black=k, brown=n, buff=b, chocolate=h, green=

In [*]: population = str(input("Choose the type of population growth from the following: abundant=a, clustered=c, numerous=n, scatter

In [*]: habitat = str(input("Choose the type of habitat from the following: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w
```

We took these variables and placed them into a dictionary assigning each one to the proper column. We then converted this dictionary into a DataFrame and concatenated it to our first copy of the data so that we could properly convert the data into numerical data for our edibility prediction of the mushroom.

```
df = df.drop(columns = ['class', 'veil-type'])

dict = {'cap-shape' : [cap_shape], 'cap-surface' : [cap_surface], 'cap-color' : [cap_color], 'bruises' : [bruises], 'odor' :
       'gill-attachment' : [gill_attachment], 'gill-spacing' : [gill_spacing], 'gill-size' : [gill_size], 'gill-color' : [gil
       'stalk-shape' : [stalk_shape], 'stalk-root' : [stalk_root], 'stalk-surface-above-ring' : [stalk_surface_above_ring],
       'stalk-surface-below-ring' : [stalk_surface_below_ring], 'stalk-color-above-ring' : [stalk_color_above_ring],
       'stalk-color-below-ring' : [stalk_color_below_ring], 'veil-color' : [veil_color], 'ring-number' : [ring_number],
       'ring-type' : [ring_type], 'spore-print-color' : [spore_print_color], 'population' : [population], 'habitat' : [habita
       }

df3 = pd.DataFrame(dict)

df3 = pd.concat([df, df3], ignore_index = True)
df3.reset_index()

for i in df3.columns:
    df3[i] = df3[i].astype('category')
    df3[i] = df3[i].cat.codes

df3.tail(1)
```

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	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
8124	2	0	2	0	2	0	0	0	10	0	...	2	2	7	7	2	0	3	7

1 rows × 21 columns

Once the data was converted to numerical data with the extra row, we used the machine learning model to create an array of predictions for the dataset. We then took the last value from this array (which was the row we added) and printed out a message describing the inputted mushroom's edibility.

```
prediction = log_reg.predict(df3)

#Find the prediction for the row we added to the dataset which is the last row
if prediction[-1] == 1:
    text = "THE MUSHROOM IS POISONOUS!!"
    color = "#e80510"
else:
    text = "THE MUSHROOM IS EDIBLE :)"
    color = "#15d44b"

Markdown(f'<h1 style="color: {color}">{text}<h1/>') #with a little flair!
```

THE MUSHROOM IS EDIBLE :)

Machine Learning

The machine learning method I chose for this project was Logistic Regression, which is a type of supervised learning. In supervised learning, the algorithm learns from labeled data to make predictions based on known outcomes. Logistic regression is an algorithm used for binary classification, and takes independent variables, assigns them numerical values, and applies them to the sigmoid function, resulting in a probability score between two different classes, usually represented as 0 and 1. The output from the sigmoid function is typically classified as the positive class if this probability score is greater than or equal to a threshold (e.g., 0.5) and as the negative class if it's below the threshold.

To utilize this algorithm, I imported Sklearn's LogisticRegression function, and used it to train the model on the training datasets mentioned earlier.

```
log_reg = LogisticRegression().fit(X_train, y_train)
```

The goal of this project is to classify whether a mushroom is edible or poisonous. Logistic regression can do this by taking in mycological data and giving each mushroom a probability score. This score can then be used for binary classification of a fungus as either 1 or 0, with 1 being poisonous and 0 being edible.

Validation

After training the model on the training data, we tried predicting mushroom edibility using the testing data. Accuracy was then evaluated by comparing the predicted values to the correct values, which resulted in a score. This accuracy score was the most important metric we used for determining the model's performance, and the goal was to make this score as high as possible without overfitting the model to the dataset. Getting a score between 0.82 and 0.96 was the range that would classify the model as accurate.

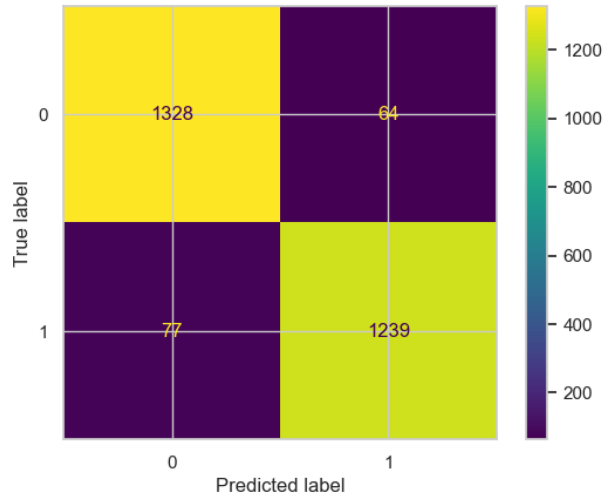
```
log_reg.score(X_test, y_test)
```

```
0.9479320531757754
```

We also visualized the overall performance of the model by using a confusion matrix, which displays the number of false positives, false negatives, true positives, and true negatives that the model predicted. Limiting the number of false positives and false negatives would be the goal here, but overfitting is also a concern.

```
ConfusionMatrixDisplay.from_predictions(y_test, y_predict_test)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x263ec312e10>
```



We also used Sklearn's classification_reports function to give the accuracy, precision, macro average, and weighted average of the model based on the test data.

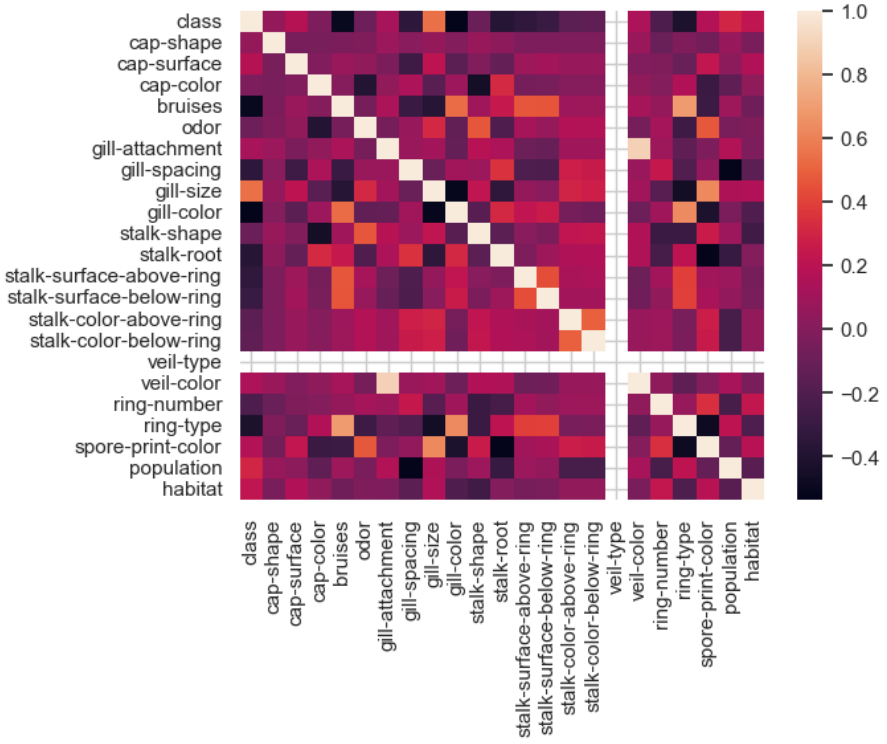
```
print(classification_report(y_test , log_reg.predict(X_test)));
```

```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      1392
           1       0.95      0.94      0.95      1316

    accuracy                           0.95      2708
   macro avg       0.95      0.95      0.95      2708
weighted avg       0.95      0.95      0.95      2708
```

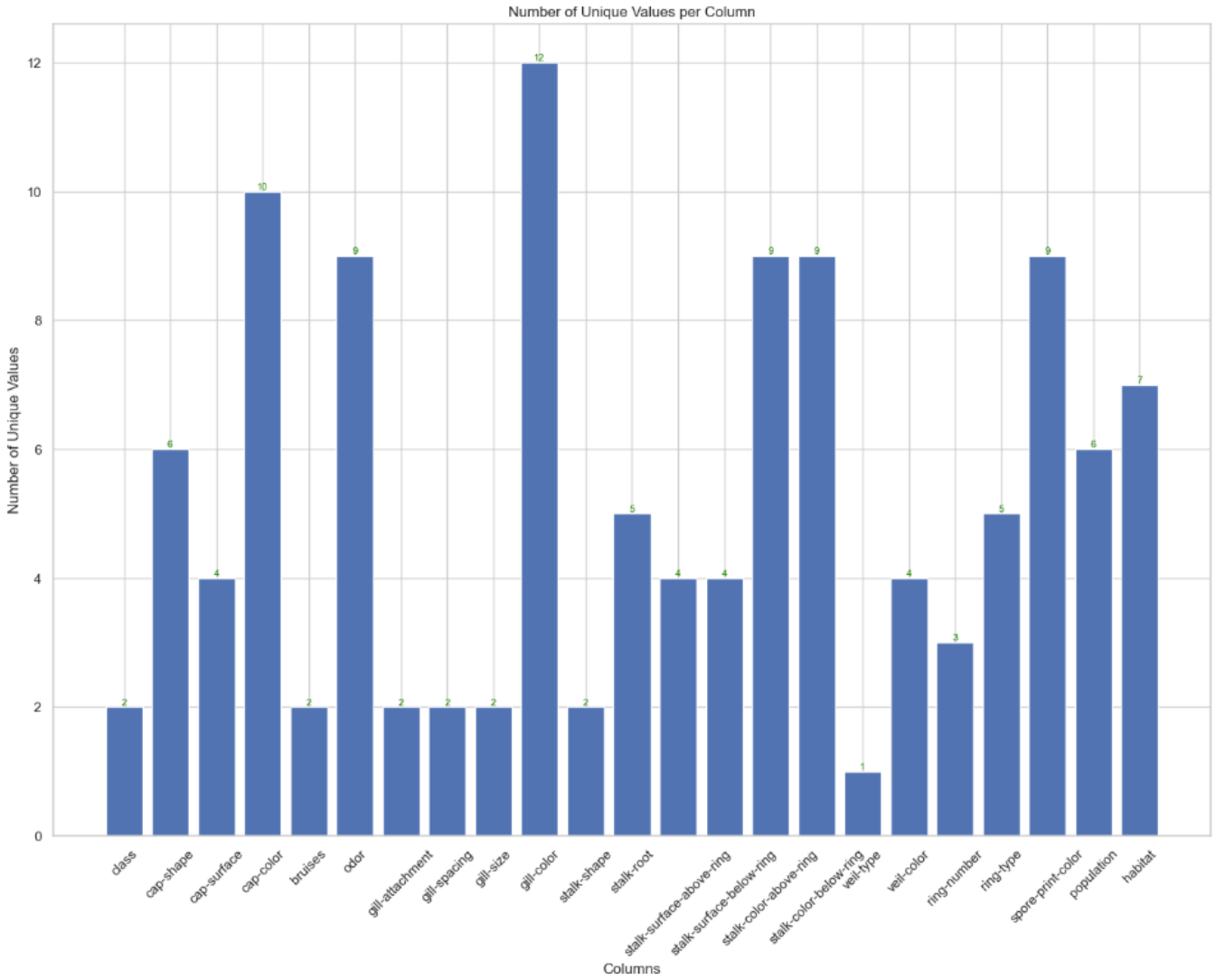
Visualizations

The following visualizations can be found within the Jupyter Notebook under the Exploratory Analysis section, except for the confusion matrix, which can be found within the Performance Report.

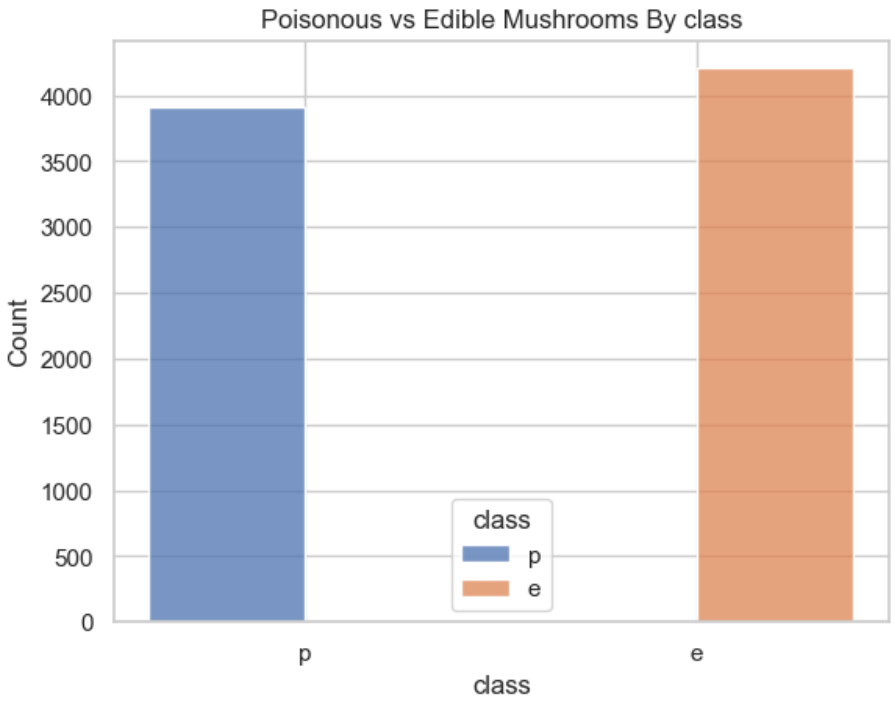
```
sns.heatmap(df2.corr());
```



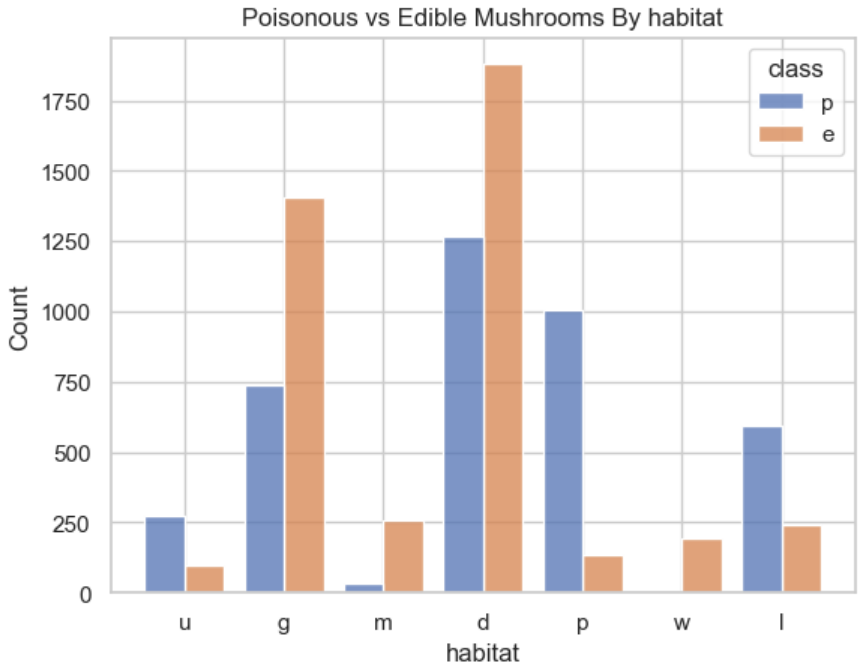
```
unique_counts = df.nunique()

plt.figure(figsize=(17, 12))
bars = plt.bar(unique_counts.index, unique_counts.values)

plt.xlabel('Columns')
plt.ylabel('Number of Unique Values')
plt.title('Number of Unique Values per Column')

plt.xticks(rotation=45)

for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2.0, yval, int(yval), va='bottom', ha='center', color='green', fontsize=8)

plt.show()
```



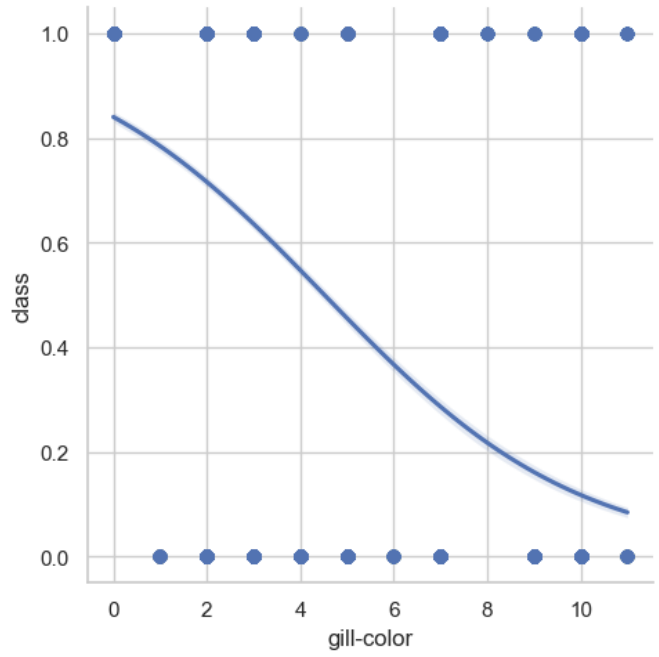
```
for i in df.columns:
    sns.histplot(data=df, x= f"{i}", hue='class', multiple="dodge", shrink=.8)
    plt.title(f'Poisonous vs Edible Mushrooms By {i}')
    plt.show()
```



...

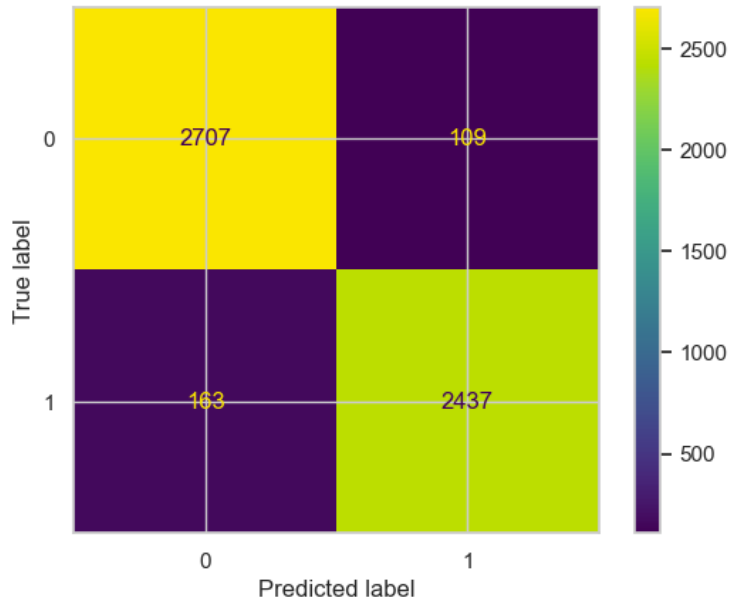


```
sns.lmplot(x="gill-color", y="class", data=df2, logistic=True);
```




```
ConfusionMatrixDisplay.from_predictions(y_train, y_predict_train)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x22b4143c490>
```



User Guide

1. Go to https://www.anaconda.com/download and download Anaconda Navigator.
2. Go through the installation process leaving the default options as they are.
3. Open Anaconda and launch Jupyter Notebooks.
4. Upload “Mushroom Edibility Classifier.jpynb” into Jupyter Notebooks.
5. Upload “mushrooms.csv” into Jupyter Notebooks.
6. Open “Mushroom Edibility Classifier.jpynb” in Jupyter Notebooks.
7. Run all the cells from “Mushroom Edibility Classifier.jpynb” in Jupyter Notebooks.
8. In the User Interface section of the notebook, input the letter out of the available options for the mycological characteristic that best describes the fungus you are trying to classify.
9. Once all cells have been run, you will be able to view the mushroom’s edibility classification at the end of the notebook.

References

No sources were used in the writing of this document.