

C964 - Computer Science Capstone

Gabriel Sabella

December 26th, 2022

	2
A1 - Letter of Transmittal	4
A2. Project Recommendation:	6
Problem Summary	6
Application Benefits	6
Application Description	7
Data Description	7
Objective and Hypothesis	8
Methodology	8
Funding Requirements	9
Stakeholders Impact	9
Data Precautions	9
Developer's Expertise	10
B1. Executive Summary	10
Problem Statement	10
Customer Summary	11
Existing System Analysis	11
Data	11
Project Methodology	12
Requirements	12
Analysis	12
Design	12
Implementation	13
Testing	13
Deployment	13
Maintenance	14
Project Outcomes	14
Implementation Plan	14
Evaluation Plan	15
Resources and Costs	15
Programming Environment	15
Environment Costs	16
Human Resource Requirements	16
Timeline and Milestones	16
D1. Post-Implementation Report	17
Project Purpose	17

	3
Datasets	17
Data Product Code	20
Hypothesis Verification	28
Effective Visualizations and Reporting	29
Accuracy Analysis	32
Application Testing	34
Application Files	34
User's Guide	36
Summation of Learning Experience	37
Sources	38

A1 - Letter of Transmittal

Gabriel Sabella
Boston, MA, 02135
gsabell@wgu.edu
November 15th, 2022

CEO Laura Hue,
Portland, OR, 97035
luara.hue@nwp.org

Dear Ms. Hue,

North West Parks has millions of visitors every year. People come from around the world for the views and pristine wildlife. A growing number of people visit your parks specifically to forage for gourmet and medicinal mushrooms. Yet your parks are home to some of the most poisonous mushrooms, and they are frequently mistaken for choice edibles, a mistake that can be fatal.

My recommended solution is to integrate an application into your existing website where visitors can enter mushroom attributes and be returned a classification: edible or poisonous. This would be accomplished by way of a machine learning algorithm trained by thousands of mushroom specimens.

This solution would be beneficial for NWP in many ways. Not only would you be showing your guests that their safety matters to this organization, you will be capitalizing on a

fast-growing hobby. Another benefit would be more traffic to your website, meaning more exposure to your existing revenue streams.

The total funding requirements will depend on whether you choose to integrate a final version of this product into your existing website. I am charging \$2500 to provide a prototype. If you do choose to include it, the data the application is based on must be updated to include specimens specific to the ecosystems found in your parks. This will require consulting with a mycologist to ensure that accurate data is collected. In total, \$20,000 will be required to build and integrate the final product.

I believe I am the right candidate to solve this problem because of the experience I have in combining software development and machine learning techniques. I am also well acquainted with the technologies nwp.org already uses, so I would be able to quickly integrate the final product.

I look forward to your feedback.

Sincerely,

Gabriel Sabella

A2. Project Recommendation:

Problem Summary

This application will prevent visitors in your park from unknowingly foraging poisonous mushrooms. It will be used by these visitors while in the field, so ease of use will be the primary focus. This application will include a user interface with the minimum number of inputs necessary for an accurate classification, which will be inferred with machine learning methods. After entering these inputs (e.g., ,cap shape, gill color, etc.), the user will be provided with a designation of “edible” or “poisonous.” These will be the initial components for the prototype sent to NWP. Many features will be added if you are inclined to integrate a final version, such as user accounts where visitors can log their classifications with photos and location. For now, this prototype will serve as a proof of concept for the classification accuracy that can be achieved utilizing these methods.

Application Benefits

This application will have two main benefits for NWP. Firstly, it will serve to reinforce the organization’s reputation for prioritizing visitor safety in order to enhance their enjoyment of the outdoors. Other companies would take an authoritarian approach of banning the activity of foraging altogether to avoid any liability issues. By taking a proactive approach, NWP will be showing visitors that their safety is important while supporting their outdoor hobby. Secondly, NWP will be capitalizing on a trending hobby that will continue to grow in the coming years. NWP will be laying the groundwork for a long-term revenue stream by attracting more guests to

pursue mushroom foraging on your parks. These guests would be using your existing website to access the classifier, so they will be exposed to your online store and other offerings.

Application Description

The data solution this application will utilize is a form of machine learning known as a Random Forest Classifier. This form of supervised machine learning model uses multiple decision trees to make inferences about data to form an accurate prediction. The reason for choosing the random forest classifier model is that it is quite accurate and is able to highlight the importance of certain features (Naviani, 2018). For instance, it may show that mushrooms with red caps and white gills are poisonous 80% of the time. This model is ideal for solving the classification problem at hand because it will infer for us which data points should be selected by the user.

Data Description

The dataset being used for this application is a public collection of several thousand mushroom species classified as edible or poisonous that contains twenty-two of their characteristics. The main benefit of this data set is the large number of species that can be compared. Another benefit is the number of mushroom characteristics included, which provides many opportunities to draw relationships between characteristics and class. The main drawback to this dataset is that it is not specific to the ecosystems present at your parks. This problem will be remedied by consulting with a mycologist that can update the data to include local mushroom species.

Objective and Hypothesis

The objective for this application is to accurately classify mushrooms as edible or poisonous using as few field characteristics as possible. The hypothesis is that there exists a combination of characteristics that can be used to classify a mushroom as edible or poisonous with 100% accuracy. If this combination of characteristics exists and can be inferred with machine learning methods, then the application can be used in the field for its intended purpose. If no such combination can be inferred, or if the combination must include a number of characteristics too large for practical use, then the final product would not be worth building.

Methodology

This application will follow the waterfall software development methodology. This methodology is ideal for this application because it is a stand-alone application and its production can be streamlined by the steps outlined in the waterfall methodology. The requirements for this application include accurately classifying mushrooms by interacting with an easy-to-use web-interface. The design phase will involve designing the machine learning model as well as the user interface. As soon as the machine learning algorithm works as intended, the simple web application will be developed. Extensive testing will be crucial for making sure that this application can be safely used as a major component of the final application.

Funding Requirements

The funding requirements for the prototype will be \$2500. If you are satisfied with the functionality of the prototype and wish to include it on your website, the final cost will be \$20,000, which includes consulting with a professional mycologist and surveying your land so that all species of mushroom present at your parks are represented.

Stakeholders Impact

This application has the potential to benefit both employees and clients of NWP. The hundreds of employees running the parks will be directly impacted because they will no longer have to worry about visitors getting poisoned and requiring immediate medical attention, resulting in a reduction in potential lawsuits. These employees will gain peace of mind, have more time for other projects, and experience an overall reduction in stress knowing visitors are safe from this threat. Executives will notice an increase in guests returning to forage, as well as an increase in new hobbyists choosing to try foraging at NWP locations due to the security the application will provide. There will be an overall increase in loyal visitors, which will benefit all NWP stakeholders.

Data Precautions

The dataset for the prototype is in no way sensitive or protected. This dataset has been around for many years and utilized in a variety of applications. If NWP chooses to integrate the final application, that dataset will be proprietary, and it is entirely the choice of the organization

what happens with that data. I believe that it would be most prudent to make said data available to the public as it could help save lives. It would be NWP's choice if they would prefer to keep that data private to ensure that nobody copies the application and takes away revenue. However, it seems that open sourcing this hypothetical data set would be most in line with NWP's culture and interests.

Developer's Expertise

I have created many full-stack applications that use machine learning. These applications have provided effective solutions and continue to improve over time. I also have years of experience deploying websites that utilize the same technology NWP currently uses for its website. With this combination of experience and skills, I will be able to quickly solve the problem at hand.

B1. Executive Summary

Problem Statement

The problem that this application will be solving is to accurately predict if a mushroom is edible or poisonous by entering readily available field characteristics. More specifically, this application aims to answer two main questions. Firstly, which characteristics are most significant for classifying mushrooms? Secondly, what is the fewest number of characteristics necessary that will yield an accurate prediction whether the specimen is edible or poisonous?

Customer Summary

The users of this application will be those visiting NWP's locations for the purpose of foraging wild gourmet and medicinal mushrooms. The application will allow them to safely forage their mushrooms knowing that they are only picking safe species. The application will be used in the field as park visitors are deciding whether to pick a specimen or leave it be. The special skills necessary will be minimal. As long as visitors accurately select characteristics, they will be given an accurate prediction. However, users must be warned that if their senses are compromised that the application cannot be safely used. For example, if a user is colorblind or has some other impairment, they should not use the application. Other than that, the application should be straightforward for hobbyists and professionals alike to use.

Existing System Analysis

NWP's website uses the python-based framework Django and the Django template system for its front end. The database for nwp.org is SQL based, specifically PostgreSQL. There are no functional gaps that this application will address, the final product would simply be integrated into the existing website.

Data

The data used for this application prototype is a CSV file with clean data. It is a collection of over eight thousand mushroom species obtained from the UCI machine learning data archive. The data classifies mushrooms as edible or poisonous and includes twenty-two of their characteristics, ranging from cap shape to habitat. In order to make this data usable, the

nominal characteristics will first need to be converted to ordinal data. This must be done first so that the data can be analyzed using machine learning methods.

Project Methodology

The development of this application will follow the waterfall methodology. This specific methodology will allow the project to be developed quickly while following an intuitive progression.

Requirements

The requirements for this application include accurately predicting whether a combination of mushroom attributes would result in specimens that are edible or poisonous, as well as using the fewest number characteristics necessary to achieve this result.

Analysis

Analysis of the data will take place in a Jupyter notebook so that the data can be visualized easily. The data will be explored by inputting different combinations of characteristics into the random forest classifier to observe results and infer key characteristics.

Design

Once key characteristics are inferred, the design phase can begin. The user interface will consist of a styled HTML form with these characteristics as options and a button to submit the results. The submitted characteristics will then be inputted into a function that will predict the class of mushrooms that match these characteristics. The images obtained during the

analysis phase will be displayed underneath this form. A brief instruction manual with example attribute combinations will also be displayed next to the form.

Implementation

The first step in implementation will be to initiate the flask application. Once the basic flask application is in place, an HTML template will be created containing a form that submits user-selected attributes back to the main python file. This python file will contain the function that predicts the class of mushrooms matching these characteristics. The prediction result along with its accuracy over the number of matching specimens will be returned and displayed in the user interface. Basic styling will then take place to make the user interface more presentable and easier to use.

Testing

Once these steps have taken place, the application will then be tested by submitting many different combinations of characteristics to see if the correct results are being returned or if the program crashes for any reason. If present, these errors will be addressed and the application will be retested.

Deployment

Once the application works as intended, it will be deployed, which will be done using the free hosting service PythonAnywhere. This service is ideal for hosting basic python applications, and will remain online for three months following the initial deployment.

Maintenance

Maintenance will be minimal for the prototype. To keep the application online, PythonAnywhere requires the author to log in every three months to extend the hosting duration at no extra cost. If this does not take place, the website will no longer be hosted. No other maintenance should be necessary for the prototype.

Project Outcomes

The tangible deliverable for the initial prototype includes a web application that utilizes the mentioned machine learning model to classify mushrooms in a user-friendly way. Since this is just a prototype, the deliverables will be limited to the temporary web application for demonstration. The intangible outcomes will result only if PNW opts to integrate a final version of the application, and would include bolstering NWP's reputation, deepening visitor loyalty, and better employee morale.

Implementation Plan

- Analyze data and perform any needed modifications
- Use descriptive techniques to visualize the data
- Infer key characteristics by way of a random forest classifier
- Test combinations of these key characteristics until 100% accuracy is achieved with fewest necessary characteristics
- Create a web application with a user interface that gathers attribute selections and runs the predictive model with the submitted combination of attributes

- Test web application to validate accuracy and address any errors, no user testing required since this is a prototype
- Deploy prototype with the free hosting service “pythonanywhere”
- Perform post-deployment acceptance testing to ensure functionality

Evaluation Plan

This application will be evaluated based on the previously mentioned criteria. If the prototype web application can accurately predict mushroom class, and does not require too many (10 or more) characteristics to do so, it will be a success. The evaluation of the final web application would be far more exhaustive, but for now the prototype will be evaluated as a proof of concept.

Resources and Costs

The resources and costs associated with this application will be \$2500 for the prototype and \$20,000 for a final product integrated into NWP’s existing website.

Programming Environment

- M1 MacBook Air running macOS Monterey
- Visual Studio Code Version: 1.73.1 (Universal) with Python extension
- Python 3.19.13
- Anaconda Navigator 2.3.2
- Jupyter Notebook 6.4.12
- Flask 1.1.2 (I had planned to use Django but Flask is compatible with Django projects and is better suited for this prototype)

Environment Costs

The environment costs will be minimal because the final application would be integrated into NWP's existing website. The prototype will be hosted via a free service. If the final product is integrated, there may be a marginal increase in hosting fees depending on how much traffic the application receives.

Human Resource Requirements

The labor costs will be most of the budget. The \$2,500 pays for my time to produce the prototype. If NWP chooses to integrate the final product, \$7,500 would cover my time to work alongside a mycologist to create a custom data set and integrate an application using this data into NWP's website. The cost to consult with a mycologist who will survey NWP locations for the mushroom species present will cost \$10,000. I will personally train the NWP development team to maintain the application, so there will be no long-term labor cost increase.

Timeline and Milestones

Milestone	Start date	End date	Duration	Dependencies	Resources
Key data points inferred	12/20/22	12/21/22	Two Days	Open source data	Self, 16 hours
Predictive algorithm is nearly 100% accurate	12/22/22	12/23/22	Two Days	Inferred key data categories	Self, 24 hours
Create UI using Templates	12/24/22	12/24/22	One Day	Sufficiently accurate algorithm	Self, 8 hours
Prototype Deployed	12/24/22	12/24/22	< One Day		Self, 2 hours
Final Version created and integrated	TBD	TBD	TBD	Mycologist, NWP dev team	Self, NWP dev team, Mycologist(s)

D1. Post-Implementation Report

Project Purpose

The purpose of this application was to demonstrate to NWP that machine learning can be used to help keep their visitors safe while foraging for mushrooms. This prototype was meant to prove that NWP can keep their guests safe if they choose to create a final version of the application with a proprietary data set created by collaborating with mycologists. This application successfully demonstrated that running a machine learning algorithm on the five

most significant characteristics for classification reveals whether that mushroom is edible or poisonous can be predicted with 100% accuracy.

Datasets

The data set used for this application is a collection of over eight thousand mushrooms along with their various attributes. I created a Jupyter notebook to visualize the data. After creating a pandas dataframe with the csv data, I displayed the first five rows:

```
df = pd.read_csv("mushrooms.csv") # Initiating data frame
df.head()
```

✓ 0.4s

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...
3	p	x	y	w	t	p	f	c	n	n	...
4	e	x	s	g	f	n	f	w	b	k	...

5 rows x 23 columns

The data had to be altered in order to work with predictive algorithms. The letters representing each attribute were converted to numeric values using the Scikit-Learn Labelencoder function:

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

✓ 0.9s

	class	cap- shape	cap- surface	cap- color	bruises	odor	gill- attachment	gill- spacing	gill- size	gill- color	...
0	1	5	2	4	1	6	1	0	1	4	...
1	0	5	2	9	1	0	1	0	0	4	...
2	0	0	2	8	1	3	1	0	0	5	...
3	1	5	3	8	1	6	1	0	1	5	...
4	0	5	2	3	0	5	1	1	0	4	...

5 rows x 23 columns

The data also had to be split up into two sets to work so that an algorithm could be trained and tested. One set contained all of the attributes with class removed, the other only contained the class. These sets were then split in half to create training and testing subsets. Each of these subsets contained half of the specimens.

```
X = df.drop(['class'], axis=1) # Set 1, all attributes without class
Y = df["class"] # Set 2, only class
# Data split up into training and testing groups
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=42, test_size=0.5, train_size=0.5)
```

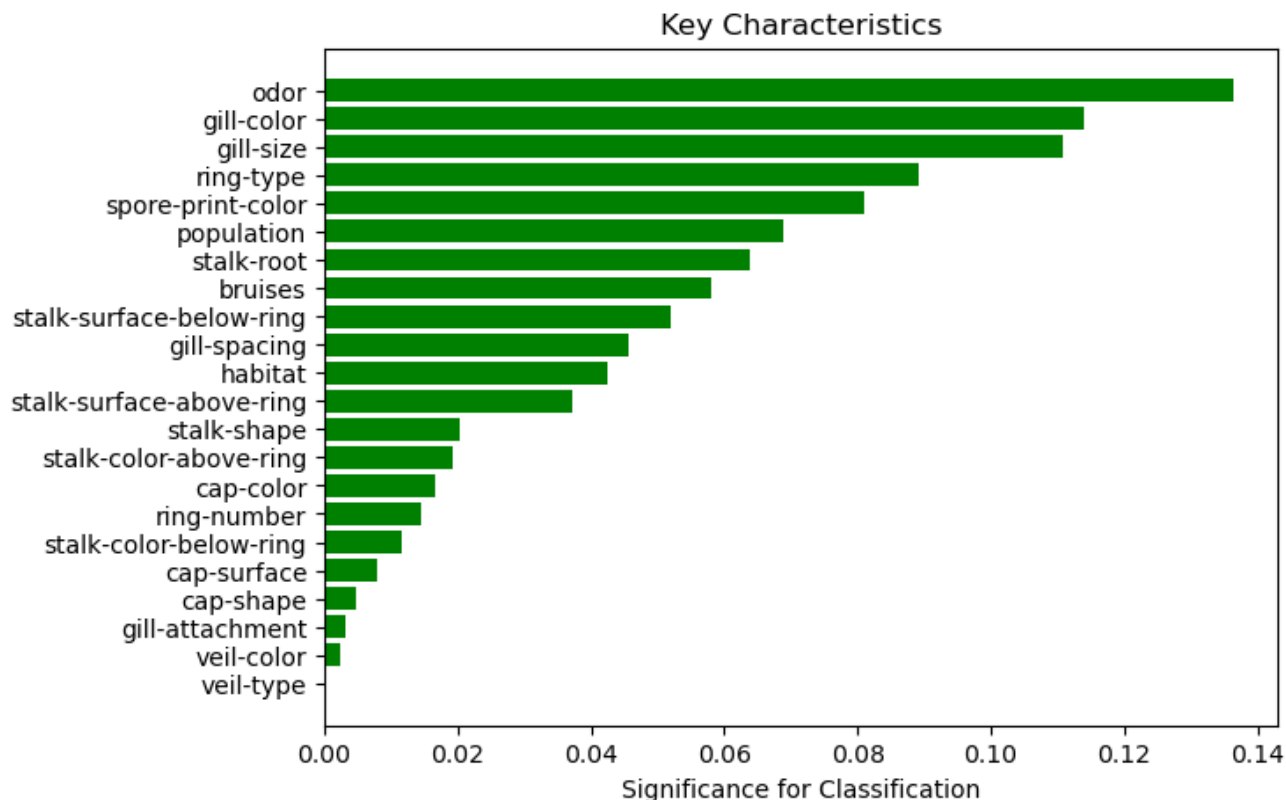
At this point the data was ready to be inputted into a random forest classifier:

```
# Initiating the Decision Tree with split data
rf = RandomForestClassifier()
rf.fit(X_train, Y_train)
```

This enabled me to use the feature importances method:

```
FI = rf.feature_importances_ # Feature Importance method
SFI = np.argsort(FI) # Sorted Based on Significance for Classification
```

I then used Matplot to visualize each characteristic ordered by their classification significance:



This visual was particularly insightful because it showed that there was only one attribute that had zero influence on classification: veil-type. Looking at the data, there are only two options for veil-type, partial and universal. Every specimen in the data set has a 'partial' veil-type. Therefore, this characteristic was removed.

Data Product Code

Now that I had compatible data with unnecessary characteristics removed, the next step was to see how accurate the random forest classifier was at predicting class. It was able to predict class with 100% accuracy being tested with all characteristics, but only once veil-type was removed. This is the highest achievable accuracy before removing veil-type:

```
Class predicted correctly 99.93 percent of the time over 4062 specimens.
```

Next I tested how well the random forest would do if it was only given a single column of data. The algorithm's scores for each characteristic are recorded in the table below:

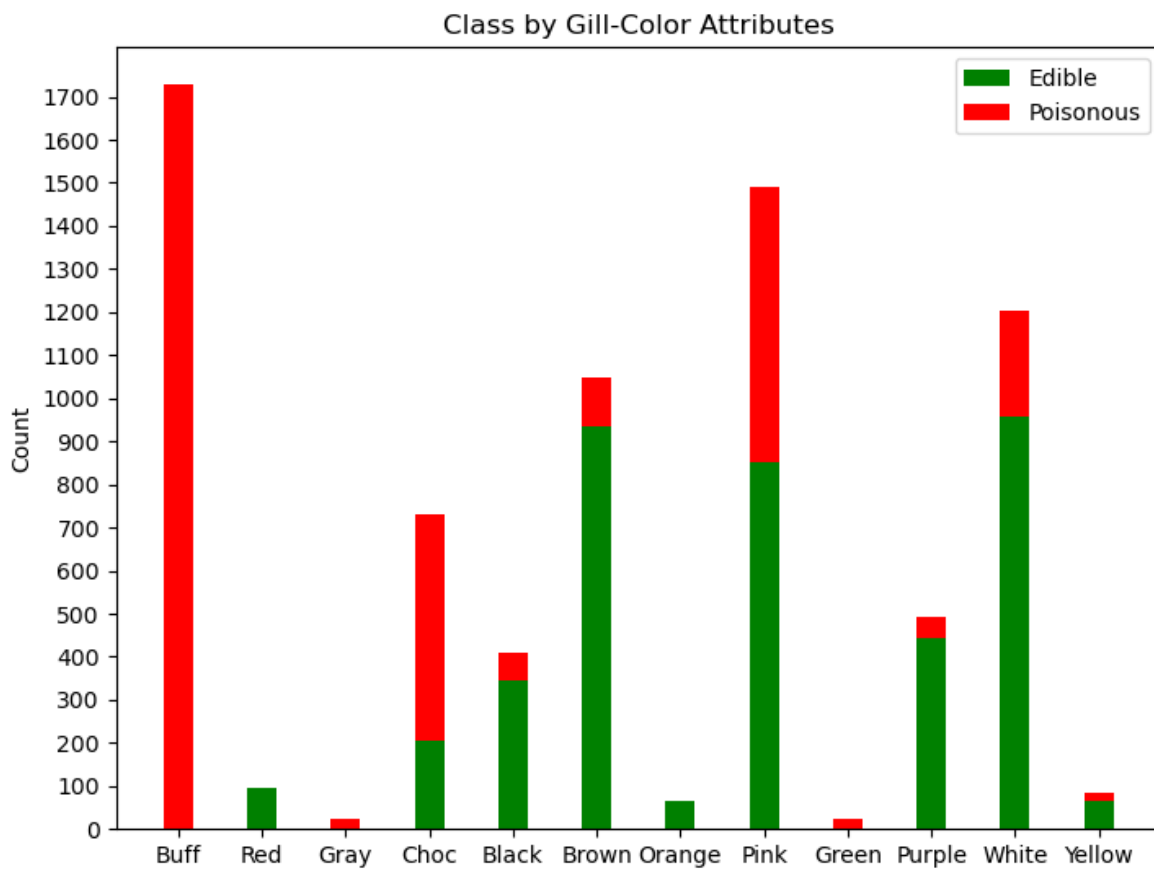
<u>Characteristic</u>	<u>Algorithm Predictive Accuracy</u>
Odor	98.25%
Gill-size	75.43%
Gill-color	80.99%
Spore-print-color	86.8%
Stalk-root	64.35%
Bruises	74.59%
Ring-type	78.31%
Population	72.62%
Cap-color	59.5%

Over 98% accuracy with just one attribute was a good sign. This meant that a combination of a few characteristics could yield 100% prediction accuracy. The results of various combinations of characteristics I tested are recorded below.

<u>Characteristics</u>	<u>Algorithm Prediction Accuracy</u>
Odor and gill-color	98.67%
Odor, gill-color and gill-size	98.7%
Odor, gill-color, gill-size, spore-print-color	99.38%
Odor, gill-color, gill-size, spore-print-color, stalk-root	99.83%
Odor, gill-color, gill-size, spore-print-color, stalk-root, bruises	100%
Gill-color and population	86.39%
Gill-color, population, spore-print-color	93.82%
Gill-color, population, spore-print-color, bruises	95.05%
Gill-color, population, spore-print-color, bruises, stalk-root	100%

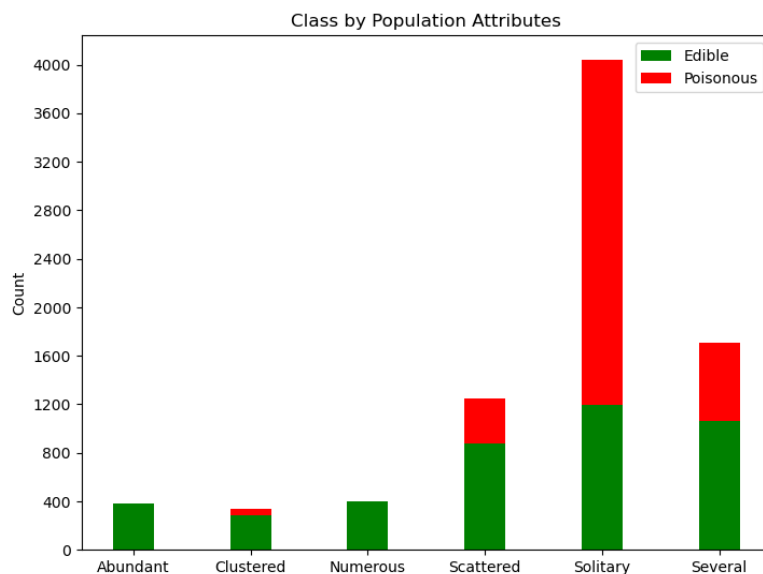
I opted to go with this final combination of characteristics to include in the user interface. This was the least number of characteristics I was able to achieve 100% predictive accuracy with. Initially I thought odor would have been included, but it took at least five other characteristics along with odor to achieve 100% accuracy.

Next, I took a closer look at all the possible attribute options within these five selected characteristics. Specifically, I used Matplot to show the quantity of specimens with each attribute and whether those specimens were edible or poisonous:

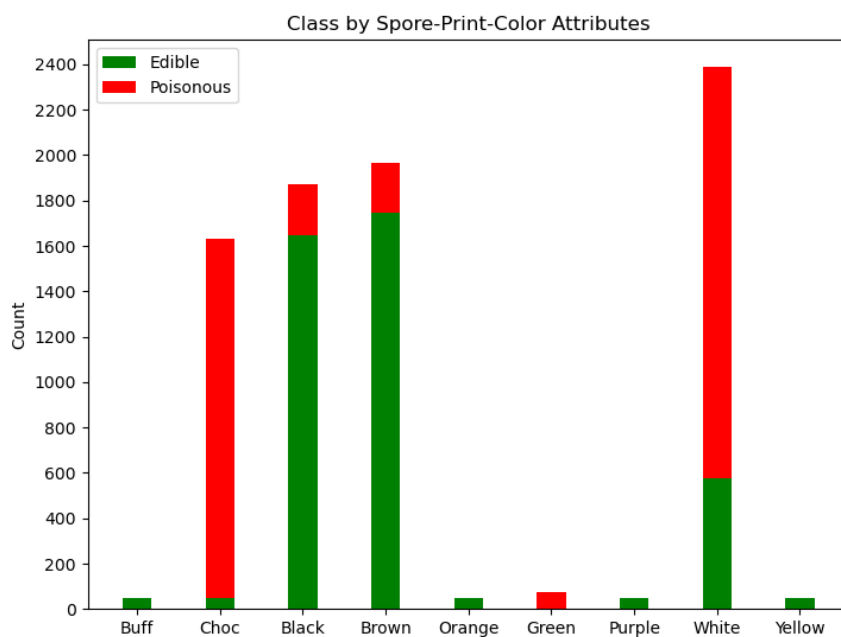


Nearly two thousand mushroom specimens have the 'buff' gill-color attribute, and they are all poisonous. This accounts for why gill-color has high classification significance.

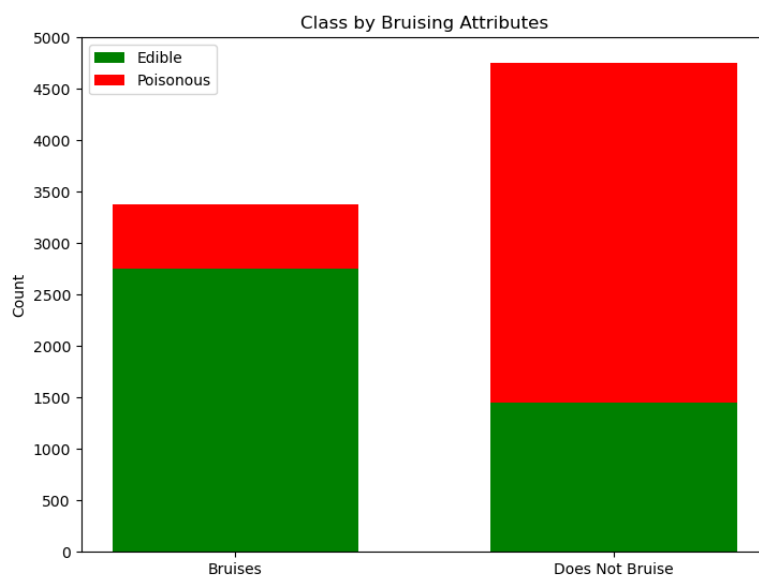
Most grouping mushrooms are edible, while solitary specimens tend to be poisonous.



Most of the spore-print-color attributes were polarized in terms of class. More than 95% of over 1600 chocolate spore-print having specimens are poisonous while specimens with brown spore prints are almost always edible.



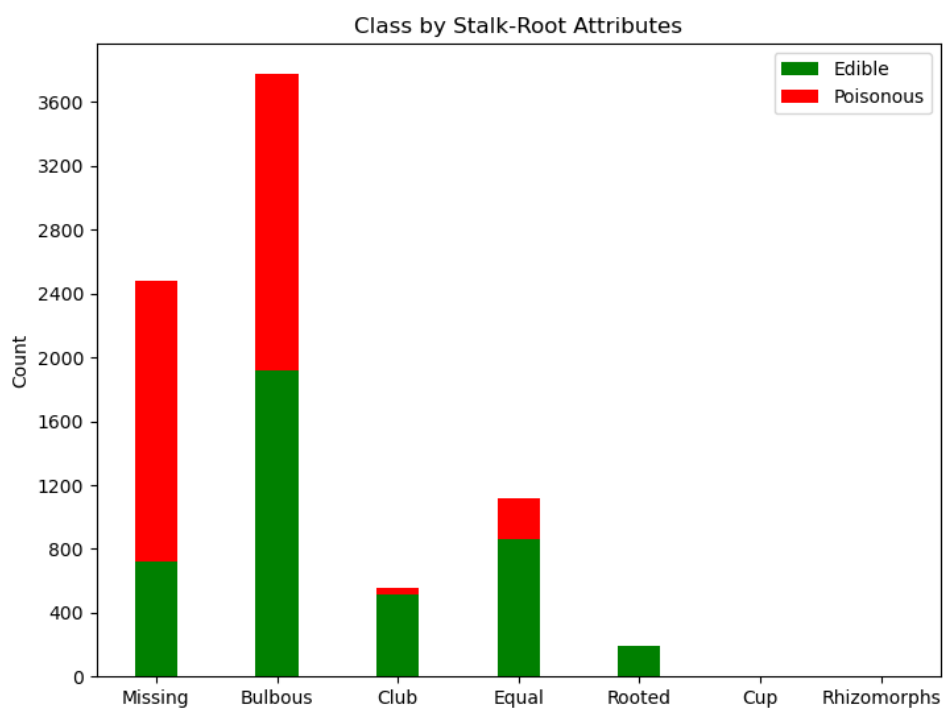
Bruising specimens were usually edible while non-bruising specimens were usually poisonous.



The bar plot below showed that the 'Cup' and 'Rhizomorphs' attributes were empty sets. I

opted to omit these options from the user interface for this prototype because they will always

lead to null results. These would be included in the final version.



At this point I had inferred the five characteristics and their attribute values that will be included in the application, so I could begin the front end. I created a simple flask application with an HTML template. This template included a form with the five characteristics as dropdown menus with attributes as options along with a button to post the results. With a bit of CSS the form looked like this:

The image shows a web form with a light gray background. It contains five vertically stacked dropdown menus, each with a label above it and a selected option shown in the dropdown box. Below the dropdowns is a blue button with white text that says 'PREDICT CLASS'.

- Gill-Color**: Buff
- Population**: Several
- Spore-Print-Color**: White
- Bruises**: False
- Stalk-Root**: Missing

Each select option was given a value that corresponds to the values within the data. These values were transferred to the main python file upon form submission.

```
<p>Stalk-Root</p>
<select id="stalk-root" name="sr">
  <option value="0">Missing</option>
  <option value="1">Bulbous</option>
  <option value="2">Club</option>
  <option value="3">Equal</option>
  <option value="4">Rooted</option>
</select>
<button type="submit" class="btn">PREDICT CLASS</button>
</form>
</div>
```

Within the main python file there was a function listening for GET and POST requests. If the method is POST, the function records the values of that form data and saves them to variables corresponding to each characteristic, shown in the following table:

Characteristic	Attribute	Value
Gill-Color	Buff	0
	Red	1
	Gray	2
	Chocolate	3
	Black	4
	Brown	5
	Orange	6
	Pink	7
	Green	8
	Purple	9
	White	10
	Yellow	11
Population	Abundant	0
	Clustered	1
	Numerous	2
	Scattered	3
	Several	4
	Solitary	5
Spore-Print-Color	Buff	0
	Chocolate	1
	Black	2
	Brown	3
	Orange	4
	Green	5
	Purple	6
	White	7
	Yellow	8
Bruises	No Bruise	0
	Bruises	1
Stalk-Root	Missing	0
	Bulbous	1
	Club	2
	Equal	3
	Rooted	4

Once captured, these values were converted to integers to be used as parameters in a function that creates a data frame with the corresponding attributes. The random forest classifier was then fitted with this data, and the results were returned:

```
@app.route('/', methods=["GET", "POST"])
def form():
    if request.method == "POST":
        gc = request.form.get("gc")
        pop = request.form.get("pop")
        spc = request.form.get("spc")
        bs = request.form.get("bs")
        sr = request.form.get("sr")
        gillColor, population, sporePrintColor, bruises, stalkRoot = int(gc), int(pop),
        int(spc), int(bs), int(sr)
        userSelection = createDataFrame(gillColor, population, sporePrintColor, bruises, stalkRoot)
        message = classify(userSelection)
        return message
    return render_template("form.html")
```

This function created the data frame from user selected values:

```
def main(gc, pop, spc, bs, sr):
    keys = kc.loc[
        (kc['gill-color'] == gc) &
        (kc['population'] == pop) &
        (kc['spore-print-color'] == spc) &
        (kc['bruises'] == bs) &
        (kc['stalk-root'] == sr)
    ]
    return keys
```

This data frame was then passed into a function that first split that user selected data into train and test data, followed by fitting a random forest classifier to that prepared data. The results were then returned with a message that displays the predicted class, along with the accuracy of predictions over the number of predictions made. This function also had to account for instances where the user submitted a combination of attributes that do not match any of the specimens within the data, which resulted in value errors. To handle these errors, I wrapped the entire function in a try/except block, ensuring the function body only ran for valid submissions.

For submissions that returned no matching specimens, an error message was displayed instructing the user to try another combination of attributes:

```
def classify(userSelection):
    try:
        # Removing class from user-selected data frame
        X = userSelection.drop(['class'], axis=1)
        Y = userSelection["class"]
        # Setting up train / test data
        X_train, X_test, Y_train, Y_test = train_test_split(
            X, Y, random_state=40, test_size=.5, train_size=.5)
        rf = RandomForestClassifier(n_estimators=50, random_state=42)
        rf.fit(X_train, Y_train)
        # Accuracy score
        accuracy = round(rf.score(X_test, Y_test)*100, 2)
        preds = list(rf.predict(X_test))
        if preds[0] == 1:
            predictedClass = 'Poisonous! '
        else:
            predictedClass = 'Edible! '
        predictions = int(len(preds))
        message = "Predicted class: {}".format(predictedClass) + "Prediction accuracy was {}%".format(
            accuracy) + " correct for {}".format(predictions) + " specimens."
        if predictions > 0:
            return message
        else:
            return 'Insufficient data'
    except ValueError:
        return 'There are no matching specimens with these selected attributes. Try a different combination of attributes.'
```

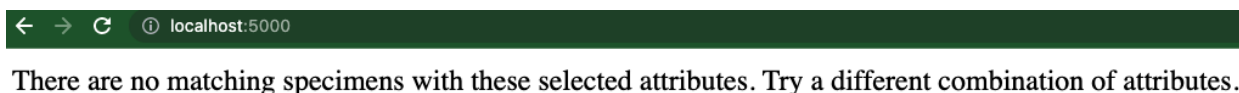
User submitted values that matched many specimens within the data:



← → ↻ ⓘ localhost:5000

Predicted class: Poisonous! Prediction accuracy was 100.0% correct for 864 specimens.

User submitted values with no matching specimens:



← → ↻ ⓘ localhost:5000

There are no matching specimens with these selected attributes. Try a different combination of attributes.

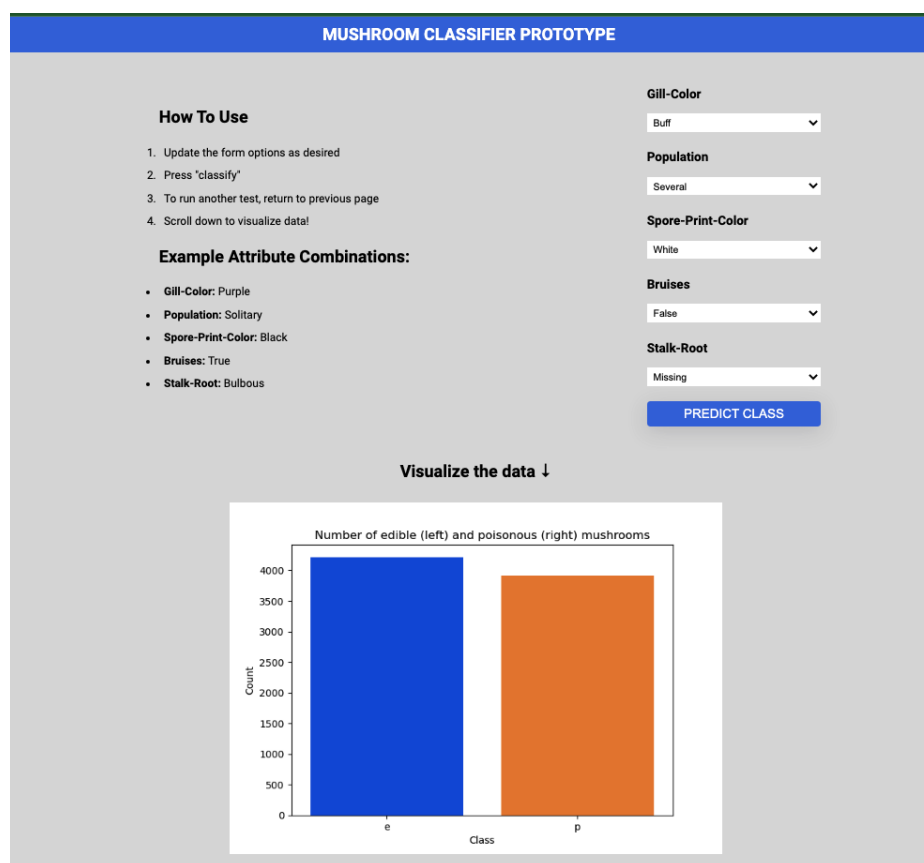
Hypothesis Verification

My original hypothesis was that there exists a short combination of characteristics that can be used to classify mushrooms with 100% accuracy. This hypothesis was accepted, although I was hoping to

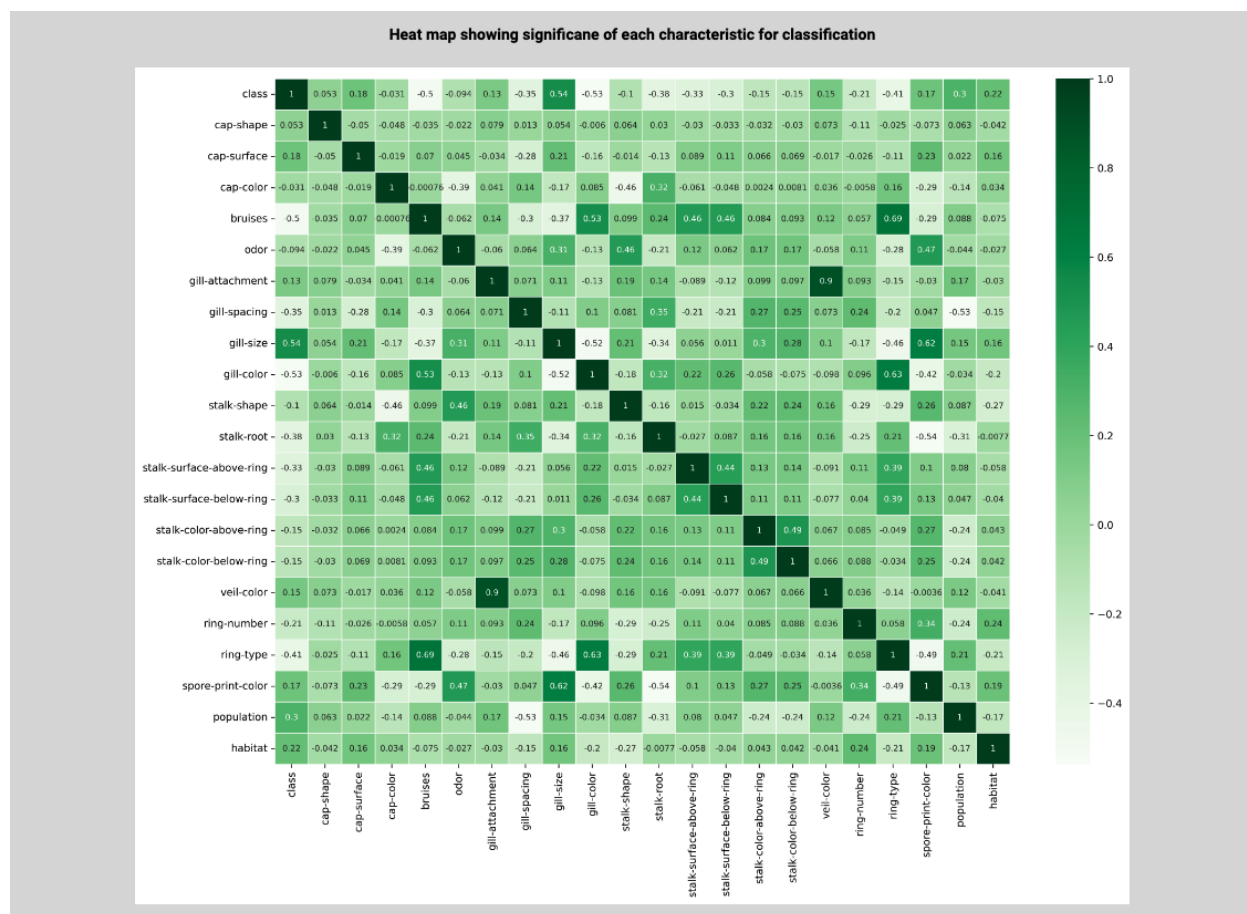
achieve this with a model trained using fewer characteristics. I showed that the characteristic odor could be used to classify all mushrooms with over 98% accuracy, but this characteristic had to be combined with five others to be fully accurate. It's possible that there is a combination of fewer characteristics that yields 100% accuracy, but the five that I tested proved my hypothesis true, and it is clear that a final version of the prototype is feasible and may be worth pursuing.

Effective Visualizations and Reporting

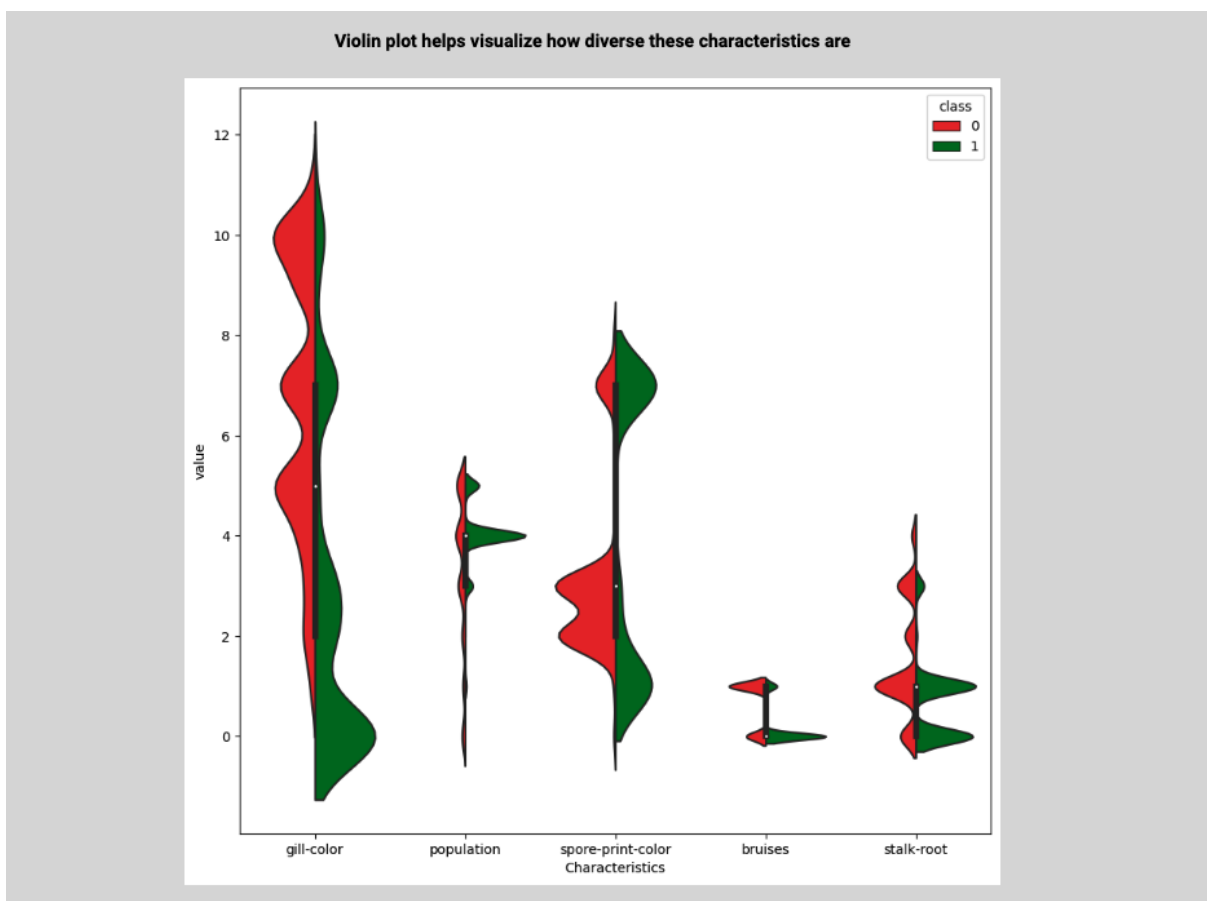
The visuals that I included in the interface serve the purpose of showing how each attribute is significant for classification as well as the number of specimens that have those attributes. The first image the user would see was a bar graph displaying the quantity of edible and poisonous mushrooms:



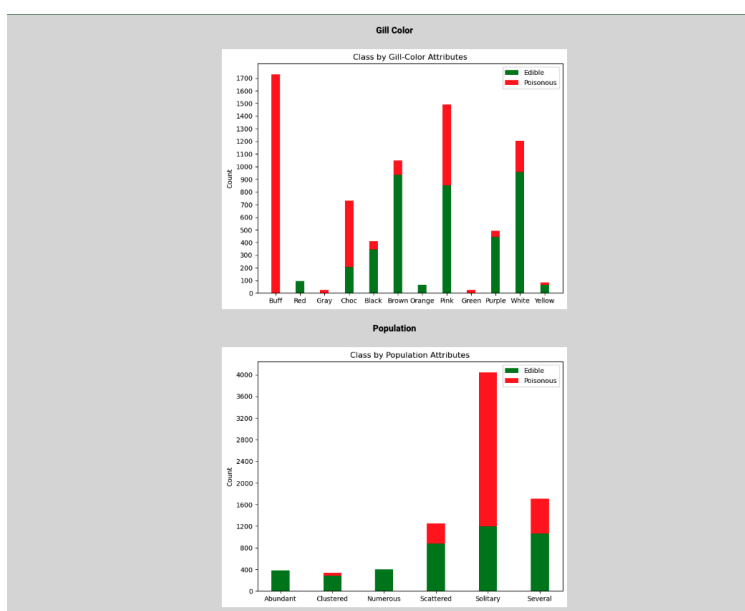
Next they would see a heatmap which showed the significance of all characteristics within the original data (except for veil-type) and their significance for classification:



The next visual was a violin plot that broke down each of the key characteristics and their attributes by class. This visual is particularly insightful because it shows how each characteristic has attributes that are especially indicative of class, both for poisonous and edible specimens.



The following visuals broke each characteristic down by attribute, showing how numerous each attribute was by class. They are the same images shown on pages 20-22:

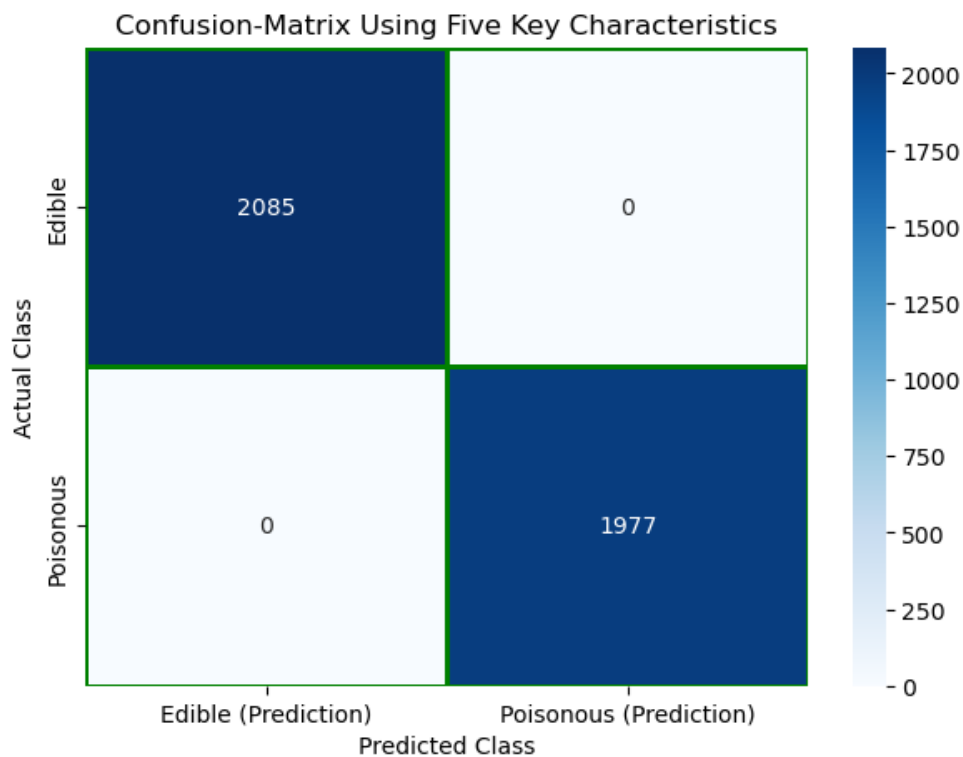




These visualizing methods were key in choosing which combinations of characteristics to test against the random forest classifier. They also enabled me to remove any extra data that harmed classification accuracy or was an attribute option without any matching specimens.

Accuracy Analysis

This prototype was able to classify mushrooms with 100% accuracy. This is because the combination of characteristics that were selected always lead either to edible or poisonous mushrooms, but never both. To visualize this I created a confusion matrix. No misclassifications are made with these characteristics:



I also used the scikit-learn classification report to verify accuracy. This report further verifies the accuracy, showing that not a single mushroom was misclassified:

```
predictions = rf.predict(X_test)
cr = classification_report(Y_test, predictions)
print(cr)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2085
1	1.00	1.00	1.00	1977
accuracy			1.00	4062
macro avg	1.00	1.00	1.00	4062
weighted avg	1.00	1.00	1.00	4062

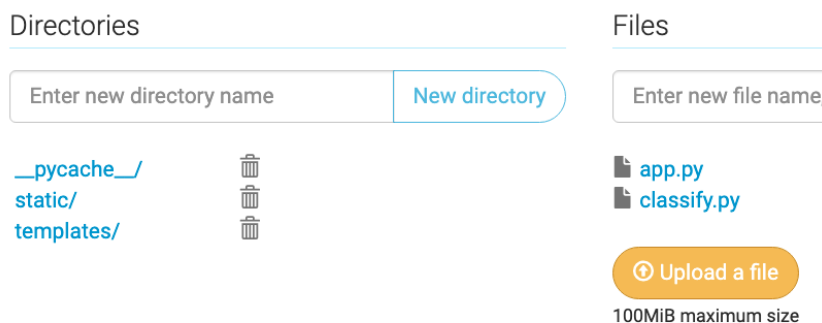
Application Testing

Unit testing was performed while adjusting the combination of characteristics being tested by the algorithm. To be in line with my initial hypothesis, I had to make sure that the algorithm was nearly 100% accurate with the fewest number of characteristics. This required testing many combinations to discern which features should be included. Once this was achieved, I had to perform usability testing to see how the application would handle submissions that did not match any specimens. These types of null results caused an internal server error, so I added an exception handler to display an informative message to the user. These tests were repeated post-deployment to ensure the application functioned as intended.

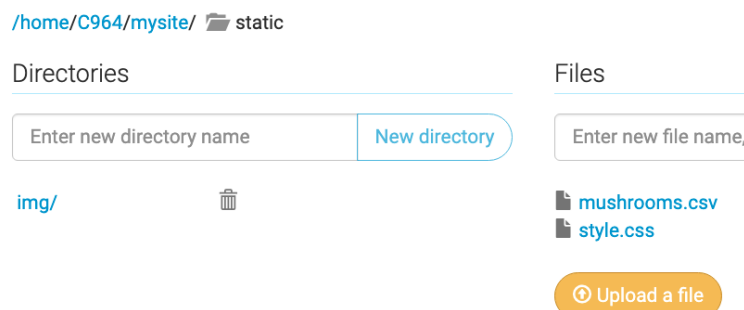
Application Files

No files are necessary to run this application. It is hosted at c964.pythonanywhere.com, simply enter the username and password to run predictions on various combinations of attributes.

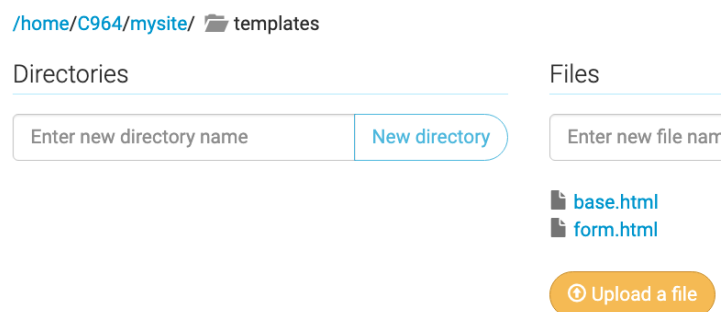
Folder structure on pythonanywhere:



The static folder contained the csv data, the CSS stylesheet, and a folder for images:



The templates folder contained both HTML templates:



In order for the host to run with the same version of python, I created a virtual environment with the correct version installed. This environment is where pandas and scikit-learn were installed.



/home/C964/.virtualenvs/flaskk/pyvenv.cfg (unsaved changes)

```
1 home = /usr/local
2 implementation = CPython
3 version_info = 3.9.13.final.0
4 virtualenv = 20.15.1
5 include-system-site-packages = false
6 base-prefix = /usr/local
7 base-exec-prefix = /usr/local
8 base-executable = /usr/local/bin/python3.9
```

After saving these changes, the website was successfully deployed, and was free from major error after testing. This hosting service includes the option to force HTTPS, which I enabled for better security.

User's Guide

The application can be found at <https://c964.pythonanywhere.com/>. Once there, enter the attributes you wish to test then press the “Predict Class” button. Here are a list of example attribute combinations that return meaningful predictions:

Gill-Color	Population	Spore-Print-Color	Bruises	Stalk-Root
black	abundant	black	false	equal
purple	solitary	black	true	bulbous
yellow	clustered	yellow	false	missing
chocolate	abundant	black	false	equal
pink	several	black	true	bulbous
white	scattered	brown	true	club

The descriptive methods can be viewed by scrolling down to where the user can visualize characteristics and their various attributes with respect to class. After making a prediction, simply return to the previous screen to run the algorithm with another combination of attributes.

Summation of Learning Experience

This capstone required me to use many technologies I had no experience with. I had never performed any sort of data analysis with python. The majority of assistance I required was fulfilled by reading the documentation for each technology, as well as referencing stack overflow posts to fix common errors. Overall the experience of working on this capstone project has reinforced for me that large, complex, and often intimidating problems can always be broken down into smaller, more approachable sub-tasks. It has also proven to me how vast the scope of problems that can be solved through programming is.

Sources

Schlimmer, Jeff (1987), Mushroom Data Retrieved from

<https://archive.ics.uci.edu/ml/datasets/mushroom>

Avinash, Naviani (2018, May). Understanding Random Forests Classifiers In Python Tutorial

Retrieved from

<https://www.datacamp.com/tutorial/random-forests-classifier-python>

Adobe Communications Team (2022, March 18). Waterfall Methodology, A Complete Guide

Retrieved from

<https://business.adobe.com/blog/basics/waterfall#advantages-of-the-waterfall-methodology>