**Introduction**

Shaggy and Scooby want to branch out from brownies and will be trying to cook mushroom stew. They feel that the best mushroom stew will look great, smell great, and be of limited toxicity. As veteran experimenters, they know there is no such thing as toxic, just toxic doses. This application will be used to help classify which mushrooms should be included in their stew based on the [Mushroom Data Set](https://archive.ics.uci.edu/ml/datasets/mushroom) from the UCI Machine Learning Repository.

* The actors are Shaggy and Scooby-Doo – Consumer of stew
* System – Will use provided Jupyter notebook to view data analysis results
* Actor-System interaction consists of using the notebook to drive a personal decision to eat the mushroom stew.

**Dataset Analysis/Import**

Our dataset will be imported directly from the UCI Machine Learning Repository using Pandas. The dataset contains 8124 observations of 23 features that are descriptive of mushroom characteristics. The variables included are:

* edible: edible=e, poisonous=p
* cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
* cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
* cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
* bruises?: bruises=t, no=f
* odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
* gill-attachment: attached=a, descending=d, free=f, notched=n
* gill-spacing: close=c, crowded=w, distant=d
* gill-size: broad=b, narrow=n
* gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
* stalk-shape: enlarging=e, tapering=t
* stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
* stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s
* stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s
* stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
* stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
* veil-type: partial=p, universal=u
* veil-color: brown=n, orange=o, white=w, yellow=y
* ring-number: none=n, one=o, two=t
* ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
* spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
* population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
* habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

All variables listed above are categorical.

**Proposed Libraries**

There are a few key libraries we will be using in this application:

* Scikit learn (sourced from <https://pypi.org/project/scikit-learn/>) – this provides our classification models
* Matplotlib (sourced from <https://pypi.org/project/matplotlib/>), Seaborn (sourced from <https://pypi.org/project/seaborn/>) – these are our plotting/visualization libraries
* Pandas (sourced from <https://pypi.org/project/pandas/>) – this provides support for dataframe manipulation
* Pandas-Profiling (sourced from <https://pandas-profiling.ydata.ai/>) – this will be used to generate a Pandas Profile for EDA
* Numpy (sourced from <https://pypi.org/project/numpy/>) – this provides functionality for numerical manipulation of arrays/Series
* Patool (sourced from <https://wummel.github.io/patool/>) – winrar archive extraction tool
* Wget (sourced from <https://pypi.org/project/wget/>) – http request for data

The application overall will be written using Python3.

**Proposed Solution/Outputs/Visualizations**

As you can see in the description of the dataset above, it does not come with any obvious target column that can be used to determine whether or not a mushroom should be included in the stew. Due to this, we will need to begin by determining a set of rules for when a mushroom should be included based on its various characteristics, and then labelling our dataset based on that. The rules we have decided on, in order of priority, are:

1. If edible=p, do not include
2. If odor=a, f, or m, do not include
3. If ring-type=f or e, do not include
4. If spore-print-color=r, o, u or y, do not include
5. If habitat=w or u, do not include
6. If cap-shape=k, f, or s, do not include
7. If cap-surface=y, do not include
8. If cap-color=r, p, u, e, or y, do not include
9. If bruises?=t, do not include
10. If gill-color=r, o, p, u, e, or y, do not include
11. If stalk-surface-above-ring=y, do not include
12. If stalk-surface-below-ring=f or y, do not include
13. Else, include

Once we have labeled our dataset per these rules, we can begin our analysis. We will first start with EDA to examine the distribution of our columns and newly created target variable (stew\_in), as well as explore the relationships between them.

Then, we will begin creating our models. The models we will explore are a Decision Tree, Random Forest, Support Vector Classifier, and Naïve Bayes Classifier.

After we create each model, we will use a confusion matrix to determine the accuracy of the classification.

**Mathematical & General Definitions**

*Decision Tree Classifier*

A Decision Tree is a machine learning algorithm that uses a set of rules to make decisions, similarly to how humans make decisions. This [requires little data preparation, while other techniques often require data normalization, dummy variables to be created and blank values to be removed.](https://scikit-learn.org/stable/modules/tree.html) Decision trees tend to overfit on data with a large number of features. Getting the right ratio of samples to number of features is important, since a tree with few samples in high dimensional space is very likely to overfit. The number of samples required to populate the tree doubles for each additional level the tree grows to.

*Random Forest*

A [random forest](https://en.wikipedia.org/wiki/Random_forest) is an estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. For classification tasks, the output of the random forest is the class selected by most trees. This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated.

*Support Vector Machine*

SVM maps training examples to points in space so as to maximize the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. SVM, optimizes the gap between margins (distance between hyper plane that separates the classes). Prediction uses -1,+1 as labels.

Definitions of results from SVM:

Where e is a point in the hyper plane,

e = 0 at least 1 unit away

0 < e < 1 - classification is correct

e >= 1 appears on wrong side and is misclassified

*Naïve Bayes Classification*

Naïve Bayes is another classification technique that uses probabilities to determine how to classify each feature. One of the keys to this method is an assumption of independence among the features. This is difficult to obtain in practice, which can lead to a higher error rate than other methods (Data Mining: Concepts and Techniques, pg. 353).

**Conclusions**

We expect a high degree of accuracy for our model results (90%+) which we will determine from the confusion matrix. We expect the Random Forest to be the most accurate model as it is an ensemble method that relies on “wisdom of the crowds”.

**Meme**

<https://giphy.com/gifs/dance-happy-dancing-Y4rBAwBrTOOggtksBK>

**Reference Links:**

<https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/>

<https://datascienceplus.com/mushrooms-classification-part-1/>

<https://datascienceplus.com/mushrooms-classification-part-2/>

<https://scikit-learn.org/stable/modules/naive_bayes.html>