EMORY
COLLEGE
OF ARTS AND
SCIENCES

ACADEMIC ARTICLE

Machine Learning

Prediction of Mushrooms Edibility

Marcos Rivas[1]* | **Rishika Shah**[1]* | **Rex Zhang**[1]*

[1]College of Arts and Sciences, Emory University, Atlanta, GA, United States

Correspondence
Email: mrivasr@emory.edu, rex.zhang@emory.edu, rishika.shah@emory.edu

Abstract

Mushroom poisoning is a problem that has been increasing in the past years. Many have experienced mushroom poisoning and have suffered severe complications in the emergency or intensive care of a hospital. The Philadelphia Inquirer reports that one of the primary reasons for the spike is that many people underestimate this problem and judge edible or poisonous from a glance. Social Media has also led to the rise of a subculture of foragers that have propagated severe cases of complications consequent to picking mushrooms in the wild. Our Machine Learning project aims to study the correlation between the many features present in a mushroom. Therefore we have built a platform that allows users to consider one of the many 118 visual features in no more than 30 inputs and outputs a prediction based on five machine learning models.

KEYWORDS

mushroom, poisoning, edibility, platform, prediction, *machine-learning*

1 | INTRODUCTION

Mushroom poisoning is a relatively common misjudged phenomenon. According to a 2018 study, there have been more than 133,700 cases of Mushroom Poisoning from 1996-2016 in the United States alone (Brandenburg and Ward). About

* Equally contributing authors.

8.6% have serious adverse health effects, and 52 have resulted in fatalities. This problem has worsened in recent years, and these numbers have spiked even more. Many people have frequented emergency and intensive care. According to the Philadelphia Inquirer, one of the primary reasons for the spike is that many people think that they can distinguish between edible and poisonous mushrooms just by looking at them, based on information and pictures they find online. Of course, these sources are only sometimes reliable. Social Media has also led to the rise of a subculture of foragers, which has increased the number of inexperienced people picking out wild mushrooms and eating them.
Through our Machine Learning project, we aim to remove human error in identifying mushroom species and move away from these unreliable sources. We attempted to achieve this by moving to a data-driven Machine Learning approach that compares several different Machine Learning models. Moreover, the Secondary Mushroom dataset from the University of California Irvine machine learning repository used for the project has more than 60000 samples from more than 170 mushroom species.

2 | BACKGROUND

This Secondary Dataset was created by simulating the Primary Dataset (populated in 1987). This initial dataset contains minimal data compared to the newer dataset. It has only $\sim$ 8000 samples from 23 species from just two mushroom samples. Much research has been done using the original dataset, and many models have been created. Through our project, we plan to build on this by using a more extensive and diverse dataset. Wagner, Heider, and Hattab in 2021 have created the Secondary Dataset.
Considering the short time since publishing this dataset, only little work has been done using this data. The only readily available research is done by the scientists that created the dataset in the first place. This project hopes to expand on the previous study by training different models, testing with varying values of hyperparameters, and evaluating using different metrics. It will also remove any possibility of conflict of interest from the research done by the data creators by providing a third-person unbiased perspective.

3 | METHODS

This section introduces the six machine learning models implemented to predict the edibility of mushrooms for this dataset. Each subsection contains the description of a model and the justification of why that model was selected.

3.1 | K-nearest Neighbors

The K-nearest Neighbors algorithm, or simply KNN, is a supervised learning classifier that uses proximity to make classifications or predictions about the grouping of an individual data point. KNN was used because not only it's a very simple, easy-to-understand, and versatile algorithm but also that it can achieve high accuracy in a lot of prediction-type problems, which fits the problem in hand - a binary classification of mushrooms being edible or poisonous.

3.2 | Decision Tree

The Decision Tree algorithm is a tree structure where each internal node represents a test on a feature, each branch represents a value, each leaf node represents a class label, and each path represents a classification/decision rule following successive choices. Decision tree was used because it is non-parametric, meaning it does not depend on

probability distribution assumptions and can potentially work on this high-dimensional secondary mushroom dataset with excellent accuracy.

3.3 | Logistic Regression

The Logistic Regression algorithm is used to transform its output using the logistic sigmoid function (Equation 1) to return a probability value which can then be mapped to some discrete classes. Logistic regression was used because if this mushroom dataset contained linearly separable classes, the algorithm would achieve exceptional performance.

$$S\left(z\right) = \frac{1}{1 + e^{-z}} \tag{1}$$

3.4 | Naive Bayes

The Naive Bayes algorithm is a probabilistic classification model based on the Bayes theorem (Equation 2). This algorithm was used for this mushroom dataset because of its fast computational time and the assumption that it'd work well if the features were independent of each other.

$$\textit{posterior} = \frac{\textit{prior} \times \textit{likelihood}}{\textit{evidence}} \Rightarrow P\left(Y|X\right) = \frac{P\left(Y\right) \cdot P\left(X|Y\right)}{P\left(X\right)} \tag{2}$$

3.5 | Random Forests

The Random Forests classifier is an ensemble machine learning method that constructs a multitude of decision trees and, for binary classification, outputs the class selected by most trees. This algorithm was used for this dataset because it is a state of the art method that reduces variance at the cost of a slight increase in bias.

3.6 | Support Vector Machine

The Support Vector Machine(SVM) classifier is a supervised learning algorithm that finds a hyperplane in an N-dimensional space that distinctly classifies the data points. SVM was selected because this secondary mushroom dataset contains high-dimensionality, and the model is versatile for that various kernel functions can be specified.

4 | EXPERIMENTS/RESULTS

4.1 | Data description

The dataset contains 61069 hypothetical mushrooms with caps based on 173 species (353 mushrooms per species). It has 21 attributes, and includes information about several physical attributes like the cap, the spores, stem, veil, etc. It also has two columns with information about the habitat and season. A detailed description of each of the attributes is given below.

- **Predicted class:**

One binary class divided in edible=e and poisonous=p (with the latter one also containing mushrooms of unknown edibility).

- **Twenty remaining features (n: nominal, m: metrical):**

  1. cap-diameter (m): float number in cm
  2. cap-shape (n): bell=b, conical=c, convex=x, flat=f, sunken=s, spherical=p, others=o
  3. cap-surface (n): fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, wrinkled=w, fleshy=e
  4. cap-color (n): brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k
  5. does-bruise-bleed (n): bruises-or-bleeding=t,no=f
  6. gill-attachment (n): adnate=a, adnexed=x, decurrent=d, free=e, sinuate=s, pores=p, none=f, unknown=?
  7. gill-spacing (n): close=c, distant=d, none=f
  8. gill-color (n): see cap-color + none=f
  9. stem-height (m): float number in cm
  10. stem-width (m): float number in mm
  11. stem-root (n): bulbous=b, swollen=s, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r
  12. stem-surface (n): see cap-surface + none=f
  13. stem-color (n): see cap-color + none=f
  14. veil-type (n): partial=p, universal=u
  15. veil-color (n): see cap-color + none=f
  16. has-ring (n): ring=t, none=f
  17. ring-type (n): cobwebby=c, evanescent=e, flaring=r, grooved=g, large=l, pendant=p, sheathing=s, zone=z, scaly=y, movable=m, none=f, unknown=?
  18. spore-print-color (n): see cap color
  19. habitat (n): grasses=g, leaves=l, meadows=m, paths=p, heaths=h, urban=u, waste=w, woods=d
  20. season (n): spring=s, summer=u, autumn=a, winter=w

4.2 | Preprocessing

1. **Handling Missing Values:**
   Missing values were replaced in the following manner:
   - Numerical Values: Missing data was replaced by the mean (average) of that column.
   - Categorical Values: Missing data was replaced by the mode (majority class) of that column.

   This code can be found in `missing_value_handling.py`, and the resulting dataset can be found in `complete_data.csv`.

2. **One Hot Encoding:**
   One Hot Encoding was performed for each of the 17 categorical variables. Here is an example with the column cap-shape, which has the following 7 categories: bell=b, conical=c, convex=x, flat=f, sunken=s, spherical=p, others=o. For this, 6 new columns were created, and assigned values as follows:

cap-shape	bell-shape	conical-shape	convex-shape	flat-shape	sunken-shape	spherical-shape
b	1	0	0	0	0	0
c	0	1	0	0	0	0
x	0	0	1	0	0	0
f	0	0	0	1	0	0
s	0	0	0	0	1	0
p	0	0	0	0	0	1
o	0	0	0	0	0	0

This process was repeated for each categorical variable, and for each coluns, $x - 1$ new columns created where x is the number of distinct categories in that column. The code for this can be found in `preprocess.py`, and the resulting dataset can be found in `encoded_data.csv`.

3. **Normalization:**
The `StandardScaler` object from the python `sklearn.preprocessing` module was used to standard scale the data. The mathematical formula for Standard Scaling is:

$$z = \frac{x - \mu}{\sigma}$$

The code for this can be found in `normalize.py`, and the resulting dataset can be found in `standard_scaled_data.csv`.

4. **Train-Test Split:**
The data was divided into Training and Testing, with the ratio 75:25. The training dataset was later used to create and train each of the models as described later. The testing dataset was used to evaluate the accuracy, F-1 score and AUROC of each of the models.
The code for this can be found in `pearson.py`

5. **Pearson Correlation:**
The Pearson correlation matrix was evaluated for both, the training and testing data. These can be visualized using python `seaborn.hashmap` as follows:

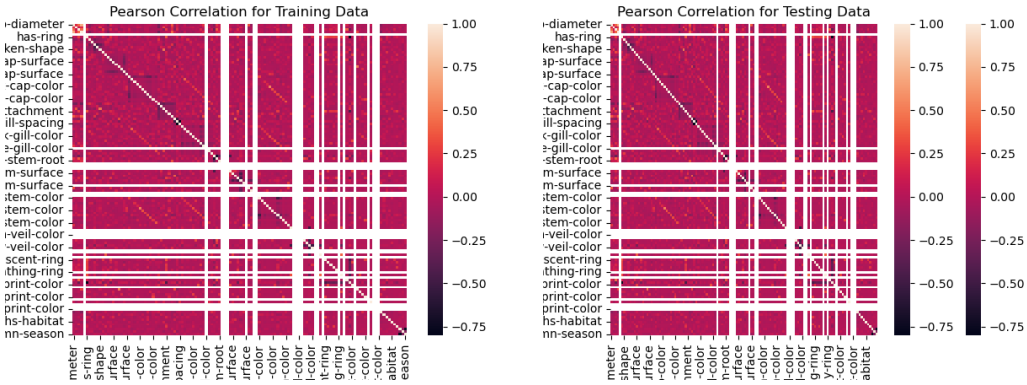

FIGURE 1 Pearson Correlation Matrices for Training and Testing Data.

All columns having Pearson correlation > 0.5 with any other column in the Training Data were dropped from both– the training and testing dataset. This was done in order to remove irrelevant features which would make the model simpler. It resulted in the following columns being dropped:

- stem-width
- none-ring
- pink-stem-color
- close-gill-spacing
- swollen-stem-root
- white-stem-color
- white-veil-color
- autumn-season

Finally, the complete preprocessed datasets were saved as `xTrain.csv`, `xTest.csv`, `yTrain.csv` and `yTest.csv`. These were used to train and evaluate the Machine Learning models. The code for this can be found in `pearson.py`.

4.3 | Modeling choices

All models from the "Methods" section above were included in our experiment. The models include K-nearest Neighbors, Decision Tree, Logistic Regression, Naive Bayes, Random Forests, and Support Vector Machine. Even though all the models were tested and their performance evaluated, there were two assumptions to be kept in mind: Logistic regression only worked well on dataset with linearly separable classes, and Naive Bayes preferred that features were independent of each other. The rest of the models were expected to work well on this secondary mushroom dataset with no necessary assumptions needed. The reasons for selecting these models can be found in the "Methods" section. Due to the large amount of data and the high-dimensionality of the data, random hyperparameters were selected to train the models before further tuning them, and accuracy, F1-score, and AUROC were used to evaluate the performance. After the initial evaluation of models trained on random parameters, it was obvious that KNN and Random Forests were the best-performing models. Thus, hyperparameters were tuned for KNN and Random Forests.

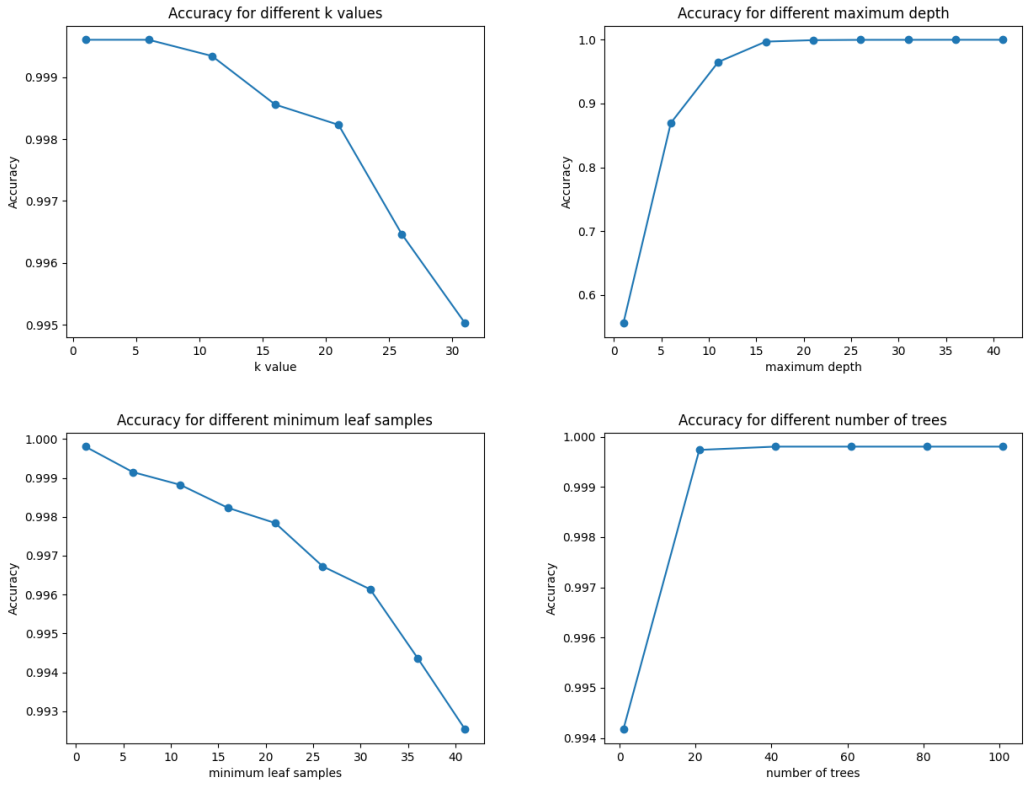

FIGURE 2 Hyperparameters tuning for the optimal k value in KNN and the optimal maximum depth, minimum leaf samples, and number of trees in Random Forests are shown in this figure.

K-nearest Neighbors: The distance metrics and the best k value were tuned for KNN. The k value was tested on the range 1 to 31, and 1 was found to yield the highest accuracy. Similarly, accuracy was used again to evaluate the best distance metrics, and Manhattan distance was determined to be the optimal distance metrics, which makes sense because Manhattan distance tends to work better than Euclidean distance when there's a high dimensionality in the data.

Random Forests: The maximum tree depth, minimum leaf samples, number of trees, and the criterion were tuned. Maximum tree depth was tested from depth of 1 to a depth of 41, and depth of 16 were found to yield the best accuracy. Minimum leaf samples was also tested from 1 to 41, and 1 was determined to be the best minimum leaf sample value based on accuracy. Similarly, the number of trees was tested from 1 to 101, and 41 was found to be the optimal number of trees by accuracy. Lastly, the best criterion was found to be gini, not entropy, based on accuracy.

Some hyperparameters tuning procedures are visualized in Figure 2. KNN and Random Forests trained on the optimal hyperparameters were implemented and evaluated while the other models were trained on random hyperparameters. The performance of all the models were evaluated using accuracy, F1-score, and AUROC, and the results can be found in the next section.

4.4 | Empirical results

Before deciding on the best machine learning model for predicting the edibility of mushrooms, it is necessary to measure the performance of the models. The classification was evaluated using accuracy with Equation (3), F1-score with Equation (4), and AUROC as the standard metrics for each of the model as mentioned in the modeling choices section. Since the class labels in the dataset were fairly balanced, accuracy was used to calculate the number of correct predictions as a percentage of the number of samples in the dataset. In addition, F1-score was also used as the harmonic mean of Precision and Recall to give a better measure of the incorrectly classified class. Lastly, AUROC was used to visualize the performance of the models based on their rate of correct and incorrect classifications.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}} \tag{3}$$

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2} \right)^{-1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The results reported using accuracy, F1-score, and AUROC are shown in Table 1, and the ROC curves are shown in Figure 3. Decision Tree, K-nearest Neighbors, Random Forest, and Support Vector Machine achieved the best results of accuracies 0.9974, 0.9996, 0.9991, and 0.9978, F1-scores 0.9965, 0.9933, 0.9991, and 0.9979, and AUROC 0.9973, 0.9996, 0.9999, and 0.9998, respectively. Logistic Regression and Naive Bayes, however, yielded sub-optimal results in comparison to the four models above. In general, the Random Forest was shown to provide the best and most consistent results for this dataset.

Model	Accuracy	F1-score	AUROC
Decision Tree	0.9974	0.9965	0.9973
K-nearest Neighbors	0.9996	0.9933	0.9996
Logistic Regression	0.8345	0.8344	0.9034
Naive Bayes	0.5891	0.5349	0.6727
Random Forest	0.9991	0.9991	0.9999
Support Vector Machine	0.9978	0.9979	0.9998

TABLE 1 Performance of 6 machine learning models on the classification of poisonous mushrooms.

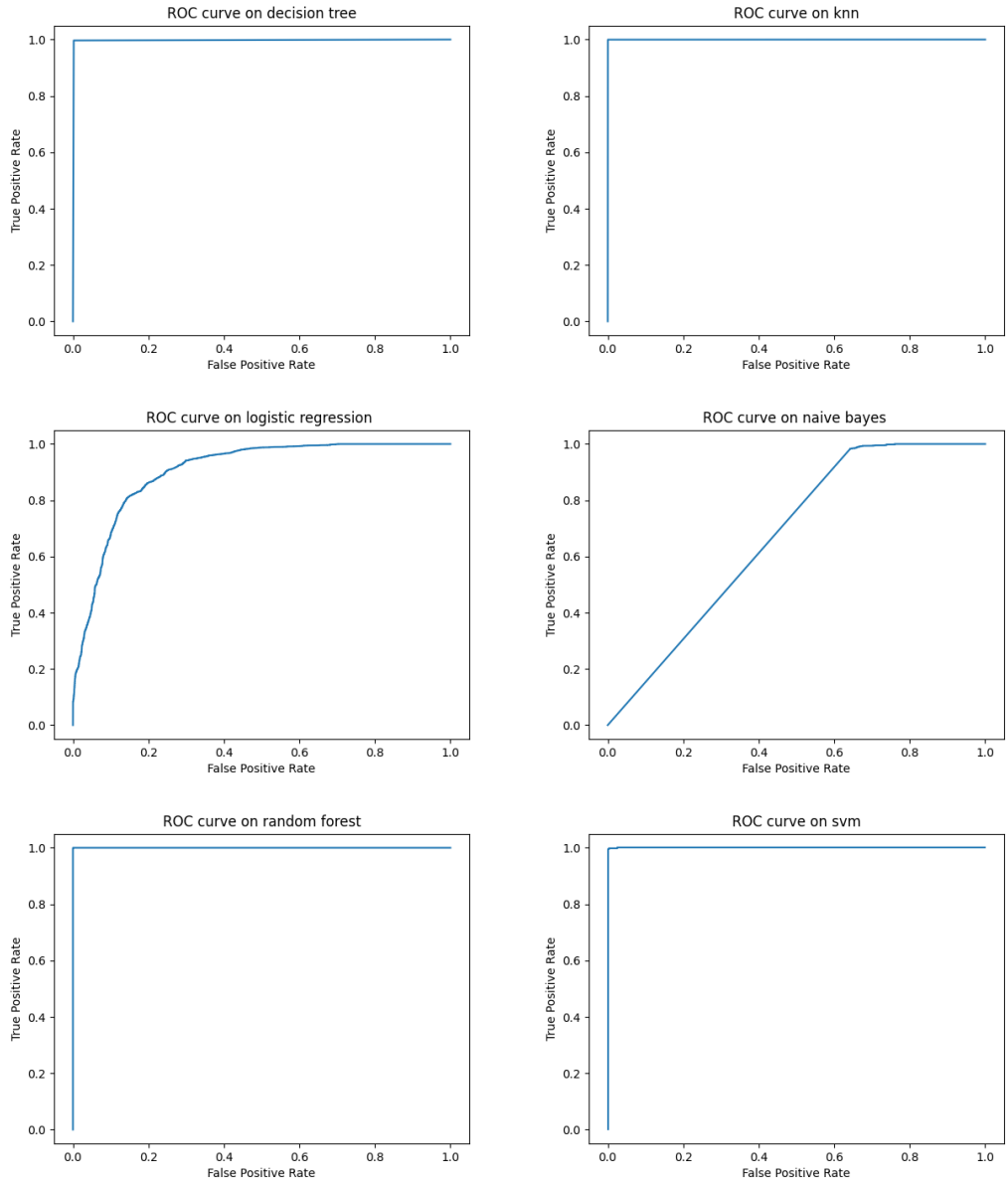

FIGURE 3 ROC curve for each classifier. The x-axis and the y-axis correspond to the FP and TP rates, respectively. While the blue lines represents the ROC curves, the areas under the curve are reported in Table 1. These curves depict the TP rates or the recall on the y-axis, and the FP rates or the Type I errors on the x-axis, which corresponds to the ratio of mushrooms wrongly classified as poisonous to all edible mushrooms.

5 | DISCUSSION

Our models for K-Nearest Neighbors, Decision Tree, Random Forest and Support Vector Machine performed exceedingly well, all having accuracy, F-1 score and AUROC $>$ 99%. We suspect the reason for this is that the dataset is extremely robust, and large with many samples. This allows our models to be trained well and thus give accurate results.

Something that can be explored in the future is how the reduction of dimensionality will affect the results. For example, we can use Principal Component Analysis (PCA) to find the first principal component. Then, we can try to train all the models after excluding that particular column. Although the accuracy will almost obviously decrease in this case, we would want to check how much it decreases by. This would be helpful to find out whether there are just one or two columns very highly correlated with the predicted class which gives the correct classification almost every time, or if it truly is the combination of these various features as we suspect.

Another aspect we would like to implement in the future is dynamic models. Through our app, we hope to collect data from users about real life mushroom samples, and their edibility status. Then, we would include these samples into the training data in order to make our models even better and more representative of real world mushroom samples.

6 | CONTRIBUTIONS

Rishika Shah was responsible for preprocessing steps of the data such as handling missing values, One Hot Encoding of categorical variables, dropping columns using Pearson correlation and Normalization. She wrote the introduction, background, data description, preprocessing steps and the discussion of the research paper.

Rex Zhang was responsible for scaling the data-set in the preprocessing part and tuning hyper-parameters as well as running and evaluating the models. For paper writing, Rex wrote the following sections: Methods, Modeling choices, and Empirical Results (figures and tables).

Marcos Rivas was responsible mainly on the product development part of our project. Marcos built an app with Flask which is a powerful tools for focusing more in the implementation part of the algorithms. The web app can be found in the GitHub repository we have included in section 7. We were able to prompt the user with no more than 20 inputs in comparison with the 118 features we have for each sample. Data is obtained and standardized in the app, then models performed and output in a table where users can look at the prediction class and score for each of the models. Marcos help edit the document and contributed to sections such as introduction and background, data description and the code section.

7 | CODE

Mushroom Prediction

In the GitHub repository above, we have included the User Friendly Interface we created to detect Mushroom's edibility and also our research progress. In the folder *Mushroom_Prepo_Models* the user can find a folder containing the raw data and another folder with the preprocessed data. Outside those folders we have included all the code we used to draw conclusions about our study.