Edible Mushroom Classification using Machine Learning Algorithms

Maissoun Ksara
Drexel University
mk3272@drexel.edu

Emmeline Pearson
Drexel University
ep623@drexel.edu

Jaspreet Singh
Drexel University
js5428@drexel.edu

Abhishek Baj
Drexel University
ab4777@drexel.edu

***Abstract*—The estimated number of mushroom species is around 1.5 million, and of those many are not well described or documented. However there are are around 2000 species that are known and documented to be edible for human consumption [8]. The non edible mushrooms can be poisonous and even lethal to consume. Classifying mushrooms by hand can be very difficult and takes a long time to master but it is useful for human health, food science, the culinary arts, and biological research. With the goal of simplifying the classification process, this study compares different machine learning methods to classify mushrooms into either edible or inedible categories. Using the UC Irvine mushroom database which has 8124 instances and 22 features representing 23 species, it was found that using an ensemble method had the best overall metrics, followed closely by decision tree and logistic regression.**

I. Introduction

Mushrooms have increasingly gained popularity in the US over the last decade, and have even been named "ingredient of the year" in 2022 by the New York Times [11]. Grocery store sales of mushrooms have increased by 20% over the last decade according to Circana, and speciality mushrooms have doubled [11]. The consumption of edible mushrooms has been linked to numerous health benefits such as improved gut and brain health, protection against obesity, type 2 diabetes, certain cancers, high blood pressure, and more [6]. They are also recognized by chefs for their ability to create savory, rich, umami flavors. However, some mushrooms are poisonous and can even lead to death if eaten [5]. Therefore, it is important to classify mushrooms as edible or poisonous for human safety and well-being.

It is estimated that 1,500,000 species of mushrooms exist in the world and only 69,000 of them have been identified to date [2]. The general methods for determining whether a mushroom is edible or poisonous are by visual identification and biochemical analysis [12]. Both of these methods require a lot of time and expertise, which many people do not have. Due to this, many research studies have been conducted that utilize machine learning, deep learning, and image processing algorithms in order to determine if a mushroom is safe to eat or poisonous [12].

In this work, we aim to classify mushrooms as edible or poisonous based on 22 different features using different machine learning algorithms using a dataset from the UC Irvine Machine Learning Repository. Logistic Regression (LR) and ID3 Decision Trees (DT) machine learning algorithms, as well as an ensemble have been used in this study for classification. Performance metrics such as the precision, recall, f1-score, and accuracy were calculated for each classification algorithm and compared to each other.

II. Related Work

In 2022, a research study was conducted by K. Kouslya et al. to classify mushrooms as edible or poisonous using four different machine learning algorithms [7]. Naive Bayes, Decision Tree (C4.5), Support Vector Machine (SVM), and Logistic Regression algorithms were employed and compared to determine which algorithm performed the best on the mushroom dataset. K. Kouslya et al. utilized LabelEncoder to convert categorical data to ordinal data without introducing differences in value ranges or losing information. They performed feature extraction and determined that gill-color feature was ranked as the most important feature, followed by spore-print-color (Figure 1) [7]. No additional information was provided regarding data preprocessing or feature extraction.

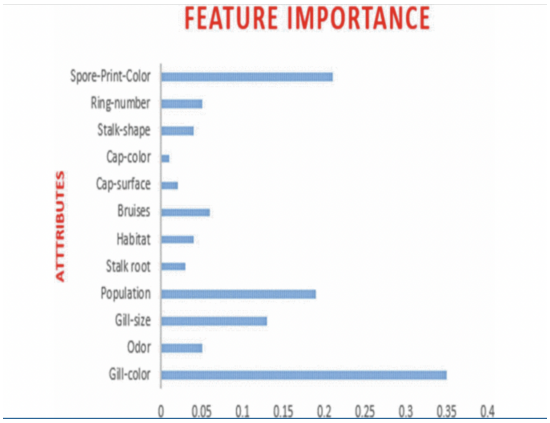

Fig. 1. Feature Importance [7]

The resulting classification accuracy of the Decision Tree was approximately 93%, followed by Support Vector Machine at 89%, Naive Bayes at 62.09%, and Logistic Regression at

86.35%, based on the dataset features. Based on their research, it was concluded that the decision tree method exhibited the highest accuracy among the compared classification algorithms [7].

In another study conducted by M. Ahmed et al. in 2022, an interpretable system for the identification of mushrooms was developed using machine learning methods and Explainable Artificial Intelligence (XAI) models [2]. They employed six machine learning algorithms including Decision Tree, Support Vector Machine, Random Forest, Logistic Regression, K-Nearest Neighbor, and Naive Bayes. They used two XAI models, SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model Agnostic Explanation), to interpret the top three machine learning models [2].

They utilized LabelEncoder to ordinalize the data and transform each value in the specified column into a numerical format. After converting the columns to the "category" type, LabelEncoder facilitated the ordinal transformation. The researchers deemed the "veil type" column, with a constant value of 0 to provide no meaningful contribution, was they removed it from the dataset. There was no information provided regarding data splitting or shuffling [2]. After the data preprocessing, M. Ahmed et al. performed the classification using the six models. The accuracy and performance metrics are shown in Figure 2 .

	Edible				Poisonous			
Model Name	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Decision Tree	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Support Vector Machine (SVM)	0.98	0.97	0.98	0.98	0.98	0.98	0.97	0.97
Random Forest	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Logistic Regression	0.95	0.96	0.94	0.95	0.95	0.94	0.96	0.95
K-Nearest Neighbour (KNN)	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**	**0.99**	**1.00**	**1.00**
Naive Bayes	0.93	0.94	0.92	0.93	0.93	0.91	0.94	0.92

Fig. 2. Performance Metrics of Mushroom data for the 6 ML Models

The top three models from these results were the Decision Tree, Random Forest, and K-Nearest Neighbor algorithms with each algorithm resulting in 100% accuracy. The SHAP values for each algorithm were determined to interpret the importance of each feature in the mushroom dataset. It was found that the gill-color, spore-print-color, gill-size, and odor are all key features shared by the three models for determining a mushroom's edibility. The SHAP values for the three models are shown in Figure 3. Although SHAP is not used in this paper, it is interesting to see what features are important based on XAI methods.

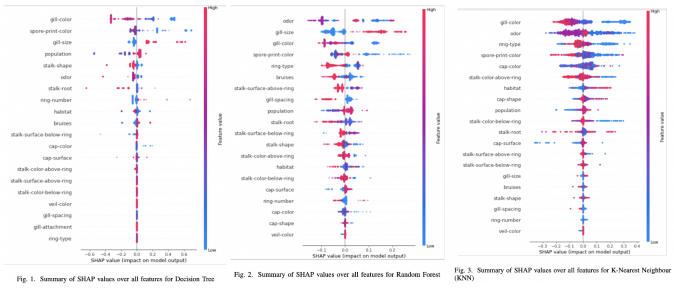

Fig. 3. Summary of SHAP values for DT, RF, and KNN

In another research study by N. Pinky et al., three ensemble methods were used to detect the edibility of mushrooms, which were bagging, boosting, and random forest [9]. The Naïve Bayes algorithm was used for bagging, AdaBoost was used for boosting, and decision tree was used for random forests. N. Pinky et al. described that out of the 22 features in the data, six of them gave a more accurate result to classifying the mushrooms. These six features were odor, spore-print-color, stalk-surface-below-ring, stalk-color-above-ring, habitat, and cap-color [9].

N. Pinky et al. created five feature sets where each feature set contains the log2N+1 number of attributes from the six features and some randomly selected features. The five features sets are:

1) Feature set-1: Odor, Cap-surface, bruises, capcolor, gill-attachment, and cap-shape.
2) Feature set-2: odor, gill-spacing, gill-size, gillcolor, spore-print-color, and population.
3) Feature set-3: odor, stalk-shape, stalk-root, stalksurface-above-ring, stalk-surface-below-ring, and stalk-color-above-ring.
4) Feature set-4: cap-color, stalk-color-below-ring, ring-number, ring-type, population, and habitat.
5) Feature set-5: cap-color, veil- type, veil-color, ring-type, population, and habitat.

They selected 2/3 for training and 1/3 for validation for the classification. After executing the three algorithms, they concluded that the model that performed the best was the random forest model using the fixed feature set, as shown in Figure 4. The models performed best using fixed models versus randomly made models [9].

		For Randomly made of models		For Fixed models	
		1/3 test	all test	1/3 test	All test
Bagging	**Naïve Bayes Based**	86.08	83.51	88.18	87.35
	Dissimilarity Measure Based	99.52	98.52	99.93	99.93
AdaBoost		99.9	99.7	-	-
Random Forest		99.15	99.26	99.93	99.93

Fig. 4. Ensemble Methods Results [9]

III. Method

A. Mushroom Dataset

The mushroom dataset was obtained from the University of California Irvine (UCI) Machine Learning Repository and contains samples which are constructed from The Audubon Society Field Guide to North American Mushrooms [13]. It consists of 8124 simulated samples representing 23 species of gilled mushrooms from the Agaricus and Lepiota genera. The samples are marked as 'e' for edible or 'p' for poisonous. If a mushroom's edibility was uncertain or unknown, it was labeled as poisonous. Of the total mushrooms, 4208 are labeled

as 'edible' and 3916 are 'poisonous', which is a fairly even split and means the data is well balanced. A bar plot of the class distribution is shown in Figure 5.

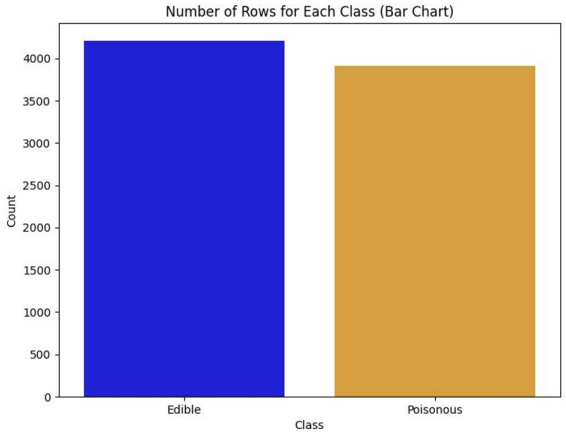

Fig. 5. Class distribution in dataset

The dataset contains 22 variables representing details about the physical, visual, and smell features of each mushroom, along with information about its habitat and growth pattern. The 22 features included in the dataset are shown below in Figure 6. To better visualize the features, Figure 7 shows a diagram of the key components of a mushroom [14].

No	Features	Nominal Values
1	Cap shape	conical (*c*), convex (*x*), flat (*f*), bell (*b*), sunken (*s*), knobbed (*k*)
2	Cap surface	grooves (*g*), scaly (*y*), smooth (*s*), fibrous (*f*)
3	Cap colour	buff (*b*), cinnamon (*c*), grey (*g*), green (*r*), brown(*n*), purple (*u*), red (*e*), white (*w*), yellow (*y*), pink (*p*)
4	Bruises	no (*f*), bruises (*t*)
5	Odour	anise (*l*), creosote (*c*), fishy (*y*), foul (*f*), almond (*a*), none (*n*), pungent (*p*), spicy (*s*), musty (*m*)
6	Gill attachment	descending (*d*), free (*f*), notched (*n*), attached (*a*)
7	Gill spacing	crowded (*w*), distant (*d*), close (*c*)
8	Gill size	narrow (*n*), broad (*b*)
9	Gill color	brown (*n*), buff (*b*), chocolate (*h*), grey (*g*), black (*k*), orange (*o*), pink (*p*), purple (*u*), red (*e*), green (*r*), yellow (*y*), white (*w*)
10	Stalk shape	tapering (*t*), enlarging (*e*)
11	Stalk root	club (*c*), cup (*u*), equal (*e*), bulbous (*b*), rhizomorphs (*z*), rooted (*r*), missing (*?*)
12	Stalk surface above ring	scaly (*y*), silky (*k*), smooth (*s*), fibrous (*f*)
13	Stalk surface below ring	scaly (*y*), silky (*k*), smooth (*s*), fibrous (*f*)
14	Stalk colour above ring	buff (*b*), cinnamon (*c*), grey (*g*), orange (*o*), brown (*n*), red (*e*), white (*w*), yellow (*y*), pink (*p*)
15	Stalk colour below ring	buff (*b*), cinnamon (*c*), grey (*g*), orange (*o*), brown (*n*), red (*e*), white (*w*), yellow (*y*), pink (*p*)
16	Veil type	universal (*u*), partial (*p*)
17	Veil colour	orange (*o*), white (*w*), yellow (*y*), brown (*n*)
18	Ring number	one (*o*), two (*t*), none (*n*)
19	Ring type	evanescent (*e*), flaring (*f*), large (*l*), cobwebby (*c*), none (*n*), pendant (*p*), sheathing (*s*), zone (*z*)
20	Spore print colour	brown (*n*), buff (*b*), chocolate (*h*), green (*r*), black(*k*), purple (*u*), white (*w*), yellow (*y*), orange(*o*)
21	Population	clustered (*c*), numerous (*n*), abundant (*a*), several(*v*), solitary (*y*), scattered (*s*)
22	Habitat	grasses (*g*), leaves (*l*), meadows (*m*), paths (*p*), urban (*u*), waste (*w*), woods (*d*)
23	Class	Edible: 4208, Poisonous: 3916

Fig. 6. Features in mushroom dataset

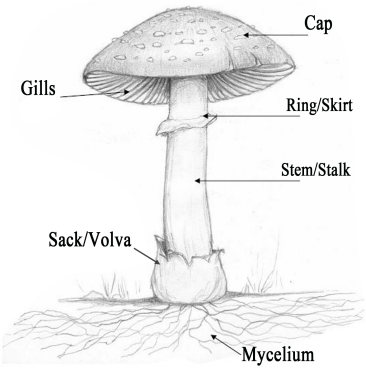

Fig. 7. Mushroom anatomy [14]

B. Data Pre-Processing

The given UC Irvine data set, contains 22 features each of which is categorical along with a target label specifying if the given sample is poisonous or edible. Each categorical feature specifies a letter to designate it's categorical feature's meaning. For example, one feature, cap-color, specifies colors such as brown denoted as n and cinnamon denoted as *c*.

1) Logistic Regression: To pre-process this data, each categorical feature letter was converted into a number starting at 0 and extending to the number of categories in each feature. The data was then randomized using a fixed seed value and split into $2/3$ for the training set and $1/3$ for the validation set. The data in each feature was then z-scored by subtracting the mean from that feature and dividing by the standard deviation.

2) Decision Tree: The preprocessing for the decision tree algorithm involved transforming the categorical variables using one-hot encoding. One-hot encoding converts categorical variables into binary vectors, in order to represent the presence or absence of a particular category. One-hot encoding was performed using pandas 'get_dummies' function. After the one-hot encoding, the data was randomized using a fixed seed value and split using the train_test_split function of sklearn into 80% for the training set and 20% for the validation set. The data was not zscored.

3) Adaboost: Similar to the above, the preprocessing for the adaboost algorithm involved transforming the categorical variables using one-hot encoding. After that, the data was randomized using a fixed seed value and split using sklearn into 80% for the training set and 20% for the validation set. The data was not zscored.

C. Logistic Regression

The first method used was Logistic Regression. Logistic regression predicts the probability that an item belongs to a class. It creates a model of how much weight to give to each feature by conduction learning iterations. After a determined number of iterations are complete, the model can be used to classify new data samples.

The algorithm for logistic regression follows the prepossessing above. Then after splitting the data set and z-scoring

the data, a bias feature was added to the x training data set. The weight values were initialized randomly and a learning rate of $n = 0.1$ was specified. The equation used for logistic regression, shown below, is based on log loss so the learning was terminated once $10,000$ epochs was reached.

$$g(x) = \frac{1}{1 + e^{-xw}}$$

After the learning was complete, the data was classified based on the weights found from the learning process.

D. Decision Tree

The ID3 (Iterative Dichotomiser 3) decision tree algorithm was the second method used. The ID3 decision tree uses weighted average entropy to make decisions. At each step, it calculates the entropy of the dataset and chooses the attribute that provides the most information gain, reducing uncertainty. This attribute is used to split the dataset, creating decision nodes in a tree structure. The process repeats recursively for each subset until a stopping point is reached. The resulting tree represents the most important features for classification, making it an effective algorithm for classifying the mushroom data.

Given probability of events $v_1, \ldots, v_K$ as $P(v_1), \ldots, P(v_K)$, the entropy of a subset can be computed as follows:

$$H(P(v_1), \ldots, P(v_K)) = -\sum_{i=1}^{K} P(v_i) \log_2 P(v_i)$$

From that, the weighted average entropy is calculated as follows:

$$\bar{E} = \frac{1}{N} \sum_{i=1}^{S} \frac{|C_i|}{N} H_i$$

where

H_i is the entropy of subset i,
$|C_i|$ is the number of observations in subset i,
S is the number of subsets created by the current feature,
N is the total number of observations.

The algorithm for decision tree followed the pre-processing discussed in the data pre-processing section. The decision tree classifier was then executed and the target values were predicted.

E. Ensemble

The process of ensemble learning involves strategically generating and combining multiple models to solve a particular computational intelligence problem. Ensemble learning is primarily employed to enhance the performance of a model, including but not limited to classification, prediction, function approximation, and other related tasks, or to mitigate the possibility of an unfortunate selection of a poor one. Other uses for group learning include valuing the decision made by the model, picking the best (or close to it), merging data, incremental learning, non-stationary learning, and correcting mistakes [10].

1) Bagging: Bagging is one of the earliest, most intuitive, and perhaps the simplest ensemble-based algorithms. In 1996, Breiman [3] found that classifiers performed well in bagging. This was achieved by using bootstrapped replicas of the training data. This means that different parts of the training data are randomly chosen and replaced from the whole set of data. Each part of the training data is used to train a different classifier of the same type. Classifiers are combined by taking a simple majority vote on their decisions. In any given case, the class selected by the most number of classifiers is the ensemble decision. Additional strategies can be employed to enhance variety, such as using a subset of the training data for preparing each classifier or using weaker methods (such as decision stumps) [10].

Algorithm 1: Bagging Algorithm

Require: Training data S with correct labels $\omega, \Omega = \{\omega_1, ..., \omega_c\}$, representing C classes
Require: Weak learning algorithm WeakLearn,
Require: Integer T specifying number of iterations.
Require: Percent (or fraction) F to create bootstrapped training data
for $t = 1, \ldots, T$ **do**
  Take a bootstrapped replica S_t by randomly drawing F percent of S.
  Call WeakLearn with S_t and receive the hypothesis (classifier) h_c.
  Add h_t to the ensemble, E.
end for
Test: Simple Majority Voting - Given unlabeled instance x
for $j = 1, \ldots, C$ **do**
  Let $v_{t,j} = \begin{cases} 1, & \text{if } h_t \text{ picks class } \Theta_j \\ 0, & \text{otherwise} \end{cases}$
  $V_j = \sum_{t=1}^{T} v_{t,j}$
end for
Obtain total vote received by each class, V_j.
Choose the class that receives the highest total vote as the final classification.

2) Boosting: Boosting creates an ensemble of classifiers by re-sampling the data, which are then combined by majority voting. Re-sampling is done to provide the most useful training data for each classifier. In simple terms, boosting creates three weak classifiers. The first one, called $C1$, is trained using a random set of training data. The training data subset for the second classifier $C2$ is chosen as the most informative subset, given $C1$. $C2$ uses only half of the training data that $C1$ correctly classified, and the other half is wrongly classified. The third classifier called $C3$ is trained on situations where $C1$ and $C2$ don't agree. A majority vote is used to combine the three classifiers [10].

Algorithm 2: Boosting Algorithm

Require: Training data S of size N with correct labels $w_i, \Omega = \{1, 2\}$

Require: Weak learning algorithm WeakLearn

1: Initialize weight $D_1(i) = \frac{1}{N}$ for $i = 1, \ldots, N$
2: **for** $t = 1, \ldots, T$ **do**
3: Train weak learner with weights D_t to get classifier h_t
4: Compute weighted error rate: $\epsilon_t = \sum_{i=1}^{N} D_t(i)[1 - h_t(x_i)]$
5: Compute alpha: $\alpha_t = \frac{1}{2} \log \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update weights: $D_{t+1}(i) = D_t(i) \exp[-\alpha_t h_t(x_i)]$
7: Normalize weights: $\sum_{i=1}^{N} D_{t+1}(i) = 1$
8: **end for**
9: **Test:** Given a test instance x
10: Compute final prediction: Use code with caution. Learn more

$$\hat{y} = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

3) Adaboost: AdaBoost algorithm works by first fitting a weak classifier on original dataset producing an output hypothesis and then iteratively re-weighting the mis-classified data to fit the next weak classifier [4]. Each weak learner is assigned a coefficient such that the sum of the training error of the resulting boosted classifier is minimized. Training a AdaBoost classifier consist of iteratively learning weak classifiers that are weighted in a way that is related to the weak learner's performance and adding them to the final strong classifier. After a weak learner is added, the input data weights are adjusted, known as "re-weighting". Re-weighting means the input data that is mis-classified would gain more weight and the correctly classified data would lose weight. Thus, the next weak learners focus more on the data that previous weak learner mis-classified. The commonly used weak learners in AdaBoost classifier are decision trees with single split called decision stumps. AdaBoost [1] refers to a particular method of training a boosted classifier. A boosted classifier is a classifier of the form.

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

where each f_t is a weak learner that takes an object x as input and returns a value indicating the class of the object. For example, in the two-class problem, the sign of the weak learner's output identifies the predicted object class and the absolute value gives the confidence in that classification. Similarly, the t-th classifier is positive if the sample is in a positive class and negative otherwise [10].

IV. Results & Analysis

A. Classification Metrics

To assess how well the different architectures performed in the classification, a confusion matrix was generated. A confusion matrix is a table that summarizes the performance of a classification algorithm based on data for which the true values are known. It consists of four values including true positive, true negative, false positive, and false negative. The diagonal values in the matrix show how accurately the models make predictions. Using these values, classification metrics such as accuracy, sensitivity, recall, precision, and the f1-score were calculated as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Accuracy} = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}}$$

Confusion matrices were generated for each algorithm using sklearn.metrics and heatmaps were created using the seaborn model in python.

B. Logistic Regression

A couple of superimposed plots of the training and validation log losses versus epoch were created after running the logistic regression training and classification. On Figure 8, the learning rate that was used was 0.1. This learning rate was too high as indicated by the oscillations in the training and validation losses. To better generalize, the learning rate was decreased to 0.01, as shown on Figure 9.

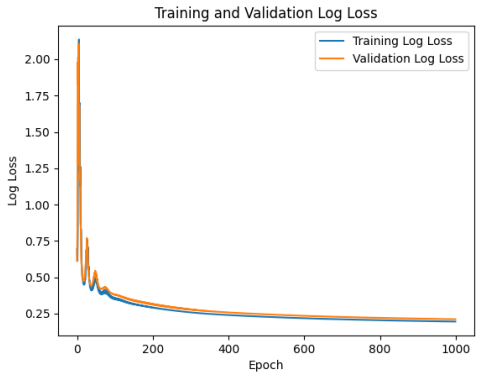

Fig. 8. Logistic Regression - Learning Rate = 0.1

Figure 9 shows a better generalization and no oscillations in the log loss. This result was used for the classifications and predicting the target values.

The result metrics are shown on Table 1. The training and validation metrics were similar to each other with slightly better values for the training data, which is to be expected.

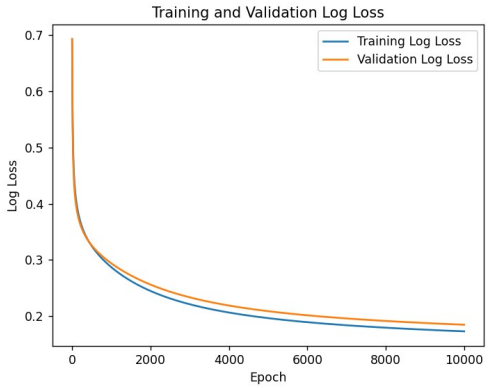

Fig. 9. Logistic Regression - Learning Rate = 0.01

Metric	Training	Validation
Precision	0.944	0.938
Recall	0.944	0.929
F-1 Score	0.944	0.934
Accuracy	0.945	0.937

TABLE 1. Logistic Regression Metrics

The logistic regression confusion matrix is shown on Figure 10. This model has a relatively high number of true positives and true negatives, indicating a good ability to predict the target class.

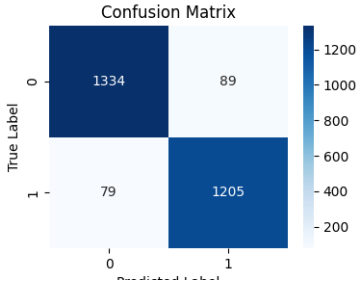

Fig. 10. Logistic Regression Confusion Matrix

C. Decision Tree

The decision tree performance metrics are shown in Table 2. Only the validation metrics were calculated. The precision, recall, f1-score, and accuracy were 1.0, 99.87%, 99.94% and 99.94% respectively. Compared to logistic regression, the decision tree algorithm resulted in higher accuracy classification metrics. Because decision trees use entropy and and are able to identify the key features, they are able to make more informed decisions on where to split. Using the best feature is key in achieving a higher accuracy and closer prediction to the target value. Based on the precision, the decision tree model was able to closely measure the ratio of correctly predicted positive instances to the total number of instances predicted as positive.

The decision tree confusion matrix is shown in figure 11. This confusion matrix shows a high level of accuracy and precision with a high number of true positives and true negatives and few errors in the model's predictions. This means the decision tree algorithm worked well with this particular dataset.

Metric	Validation
Precision	1.0
Recall	0.9987
F1 Score	0.9994
Accuracy	0.9994

TABLE 2. Decision Tree Validation Metrics

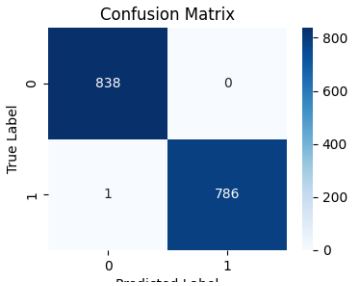

Fig. 11. Decision Tree Confusion Matrix

D. Ensemble

The performance metrics for the ensemble method are shown in Table 3. These metrics show precision, recall, and f1-scores of 1.0. The accuracy for the training and validation data were 99.35% and 99.38% respectively. Although the validation data resulted in a slightly higher accuracy than the training data, this difference is almost negligible. This high accuracy infers that the adaboost algorithm almost perfectly classified the mushroom into the correct target classes.

Metric	Training	Validation
Precision	1.0	1.0
Recall	1.0	1.0
F1 Score	1.0	1.0
Accuracy	0.9935	0.9938

TABLE 3. Adaboost Training and Validation Metrics

As shown on the confusion matrix in Figure 12, there is a high number of true positives and true negatives, meaning the model performed effectively with the dataset.

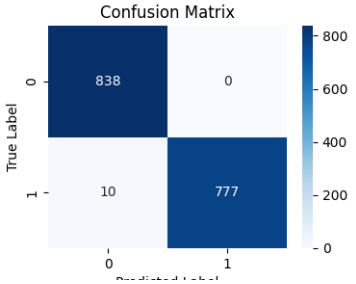

Fig. 12. Adaboost Confusion Matrix

V. Conclusion

Overall, the three methods we applied (logistic regression, decision tree and ensemble) all had positive metrics. When comparing overall accuracy logistic regression had the lowest accuracy at 0.945 for training data and 0.937 for validation data. The decision tree accuracy was 0.999 which was very was similar to the ensemble accuracy of 0.993 for validation data sets. In terms of precision, decision tree and ensemble both have metrics of 1.0 and logistic regression is slightly lower at 0.944 for the validation data. For recall logistic regression had the lowest value of 0.929 followed by 0.998 for decision tree and 1.0 for ensemble. F-1 score follows a similar pattern, starting with 0.934 for logistic regression, then going to 0.999 for decision tree and 1.0 for the ensemble.

Putting these results into the context of previous work, prior work also found that logistic regression generally had the lowest metrics. Although, our logistic regression metrics were higher than the Kousalya study [7]. For decision trees, in the kousalya study the model had a 93% accuracy compared to this result of 99%. Lastly, although, for ensamble addabot method the Ahmed paper [2] showed similar results of 99.9% accuracy compared to this result of 99.3%.

In conclusion, each of the methods chosen made accurate predictions however, in terms of ranking across all metrics, the best was ensemble, followed closely by decision tree and lastly concluding with logistic regression.

VI. Future Work/Extensions

To build upon the results gathered here, possible future work could include adding additional machine learning methods such as support vector machine, other ensemble methods, or neural networks. It would be interesting to compare if other methods could result in better metrics than the ones observed in this study.

Also, extending the data set or utilizing an entirely new data set could be interesting as well. This data set is not exhaustive of all species of mushroom. More samples could be added to extend the broader usefulness of models generated in this study. Lastly, the UCI data set was simulated so it would be interesting to see how these models worked on real data.

References

[1] Adaboost in python - machine learning from scratch 13 - python tutorial, www.youtube.com/watch?v=wf5t4mmv5ust=519s. Accessed 13 Dec. 2023.

[2] Md Sabbir Ahmed, Sadia Afrose, Ashik Adnan, Nazifa Khanom, Md Sabbir Hossain, Md Humaion Kabir Mehedi, and Annajiat Alim Rasel. Comparative Analysis of Interpretable Mushroom Classification using Several Machine Learning Models. *Proceedings of 2022 25th International Conference on Computer and Information Technology, ICCIT 2022*, pages 31–36, 2022.

[3] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[4] Peter Freund. Bodies, disability and spaces: The social model and disabling spatial organisations. *Disability amp; Society*, 16(5):689–706, August 2001.

[5] Harvard School of Public Health. Mushrooms — The Nutrition Source — Harvard T.H. Chan School of Public Health.

[6] Health.com. Are Mushrooms Good For You? Benefits, Nutrition, and Risks.

[7] K. Kousalya, B. Krishnakumar, S. Boomika, N. Dharati, and N. Hemavathy. Edible Mushroom Identification Using Machine Learning. *2022 International Conference on Computer Communication and Informatics, ICCCI 2022*, 2022.

[8] Philip G. Miles and Shu-Ting Chang. Mushrooms : Cultivation, Nutritional Value, Medicinal Effect, and Environmental Impact. mar 2004.

[9] Nusrat Jahan Pinky, S. M.Mohidul Islam, and Rafia Sharmin Alice. Edibility Detection of Mushroom Using Ensemble Methods. *International Journal of Image, Graphics and Signal Processing*, 11(4):55–62, 2019.

[10] Robi Polikar. Ensemble learning. *Scholarpedia*, 4(1):2776, 2009.

[11] The Guardian. A mushrooming trend: how fungi became an It food — Fungi — The Guardian.

[12] Kemal Tutuncu, Ilkay Cinar, Ramazan Kursun, and Murat Koklu. Edible and Poisonous Mushrooms Classification by Machine Learning Algorithms. *2022 11th Mediterranean Conference on Embedded Computing, MECO 2022*, 2022.

[13] UCI Machine Learning Repository. Mushroom - UCI Machine Learning Repository.

[14] Yellow Elanor. Mushroom Identification Basics - Yellow Elanor.