Edible and Poisonous Mushroom Classification

Project Report
Submitted in partial fulfillment of the requirements for the coursework

MCSC105 Data Mining
MSc Semester I

Submitted by

Jishnu Bandyopadhyay
(Roll No- 21, MSc Sem-I)

Deepak Jangir
(Roll No- 15, MSc Sem-I)

Under the guidance of

Dr. Bharti Rana
Department of Computer Science
(University of Delhi, New Delhi, 110009)

Introduction:

Mushroom consumption has been a traditional practice for many years, offering valuable sources of nutrition. However, the downside is that not all mushrooms are safe to eat. Some can be very toxic, and it can be difficult to tell the difference between edible and poisonous varieties. This makes identifying safe mushrooms a critical task in order to avoid serious health risks. Mushroom poisoning is a real problem globally, leading to illness and even death among both experienced foragers and people who may simply want to try picking mushrooms for fun.

Mushroom poisoning incidents can be illustrated by past events. For example, a BBC report highlighted a tragic incident in India where mushroom poisoning led to 22 deaths, underscoring the potentially fatal consequences of misidentification. In the USA, the National Poison Data System (NPDS) has been tracking mushroom exposure cases for over 30 years. The data, compiled by poison control centers across the country, provides valuable insights into the frequency and severity of mushroom poisoning. From 1999 to 2016, numerous cases were reported annually, further emphasizing the need for effective identification methods to distinguish edible mushrooms from toxic ones.

Currently, the consumption of mushrooms—both for direct eating and medicinal purposes—has increased, making the need for reliable identification tools even more important. Thanks to advances in technology, particularly in machine learning and data analysis, we now have the opportunity to develop classification models that can help people distinguish between edible and poisonous mushrooms. These tools could make a significant difference for public health by preventing accidental poisoning and making it safer for people to enjoy the many benefits of mushrooms.

Literature Review:

a. **Paper 1 (Classification and selection of the main features for the identification of toxicity in *Agaricus* and *Lepiota* with machine learning algorithms):**
This paper mainly focuses on the preprocessing part. They used various Feature Extraction methods **GALGO** and **LASSO** regression to extract the important features of the **Mushroom Dataset** from UCI Machine Learning Repository (https://doi.org/10.24432/C5959T) (id = 73).
The stalk root feature was dropped because apparently the feature is not accurately recorded for many cases, and it has 30% missing data.
LASSO Regression reduced number of features to 15 from 22. Mainly odor, gill size, stalk shape. GALGO determined the important features as odor, spore print color and habitat. They used 25% of the data for testing and remaining 75% data for training.
Then the reduced dataset is run through various classifiers, like KNN, Logistic Regression, XGBoost.

Results (Accuracy)

Feature Selection	Logistic Regression (Accuracy)	K Nearest Neighbor (Accuracy)	XGBoost (Accuracy)
LASSO	0.94	1.00	1.00
GALGO	0.64	1.00	1.00

b. **Paper 2 (IoT enabled mushroom farm automation with Machine Learning to classify toxic mushrooms in Bangladesh):**
This research utilizes the dataset retrieved from the Kaggle. The dataset is enriched with 8125 data categorized into two identical classes, namely, edible and poisonous. The dataset contains 21 characteristics of mushrooms to classify the type of mushroom. This research work represents the machine learning based model of mushroom classification system. The ml model is built with six conventional classifiers such as decision tree, logistic regression, k-nearest neighbor, support vector machine, naïve bayes, and random forest, and finds the satisfactory accuracy of classification with DT, KNN, SVM, and RF. This model also finds high accuracy of 100% with six ensemble classifiers. The study us further furnished with the system usability scale (SUS) to track the regular user's satisfaction.
After successfully retrieving the data from the farm, the data cleaning approaches will be applied to clean the dataset's null values. After that, the data will be sent for testing. The prebuilt model will provide the decision based on the tested data. The model provided the dataset into two significant classes such as edible and poisonous.

Results (Accuracy)

	SVM	Naïve Bayes	Logistic Regression	KNN, DT, RF
Accuracy	0.93	0.91	0.93	1.00

Dataset Details:

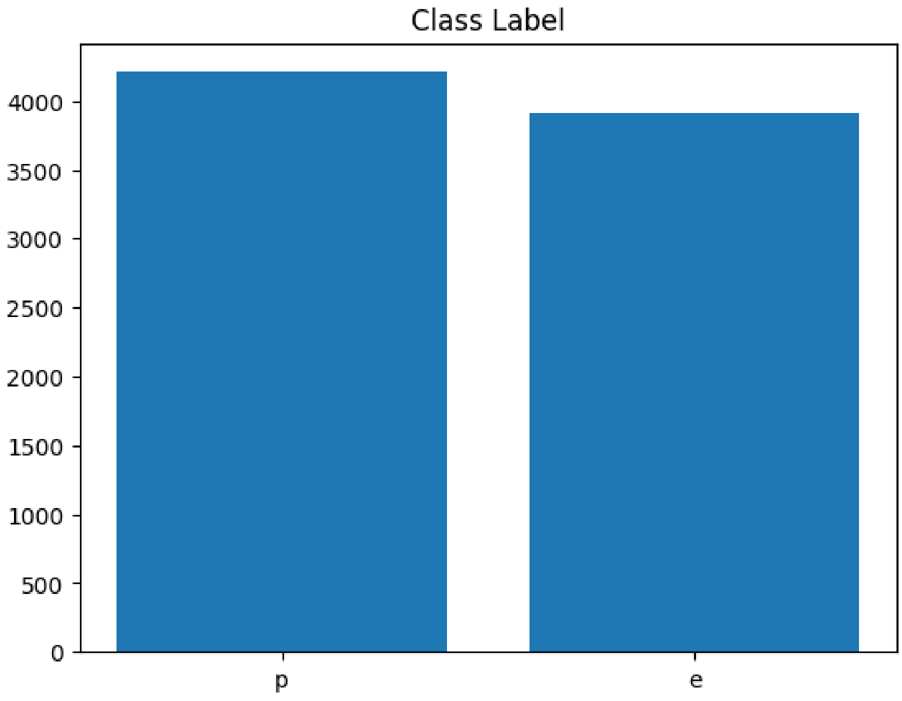
Name: Mushroom Database
Source: Mushroom records drawn from The Audubon Society Field Guide to North American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred A. Knopf
Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)
Hosted on: UCI Machine Learning Repository (id = 73) https://doi.org/10.24432/C5959T
Number of Samples: 8124
Features: 22 (All Categorical)
Labeled: All samples are labeled (Binary)
Attribute Information:

1. cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
3. cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. bruises: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. 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
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p, red=e, white=w, yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p, red=e, white=w, yellow=y
16. veil-type: partial=p,universal=u

17. veil-color:	brown=n,orange=o,white=w,yellow=y
18. ring-number:	none=n,one=o,two=t
19. ring-type:	cobwebby=c,evanescent=e,flaring=f,large=l, none=n, pendant=p, sheathing=s, zone=z
20. spore-print-color:	black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21. population:	abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22. habitat:	grasses=g, leaves=l, meadows=m, paths=p, urban=u,waste=w,woods=d

Project Methodology:

1. **Preprocessing**- In the preprocessing step, first the distribution of the dataset is visualized. The dataset has 8124 samples. Out of them, 4208 samples are labeled poisonous (51.80%). Rest of the samples are labeled edible (48.20%). So the dataset is evenly distributed, there is no class imbalance. So there is no need for undersampling or oversampling.

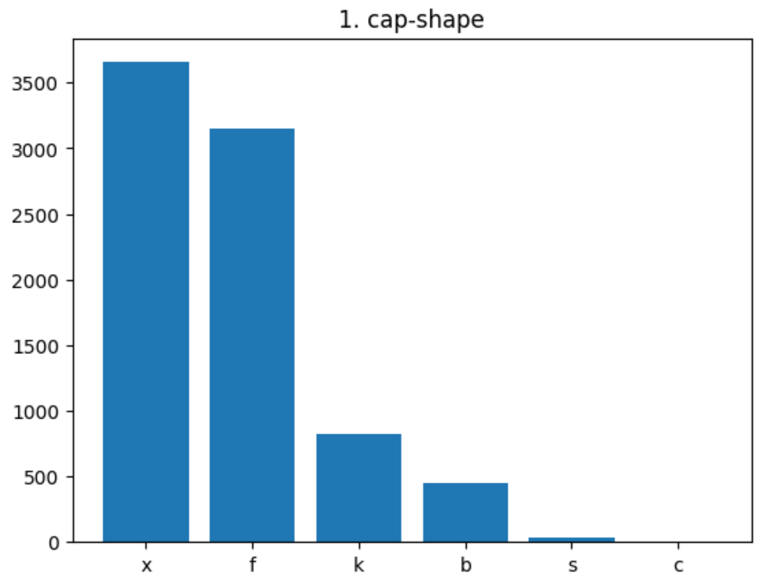


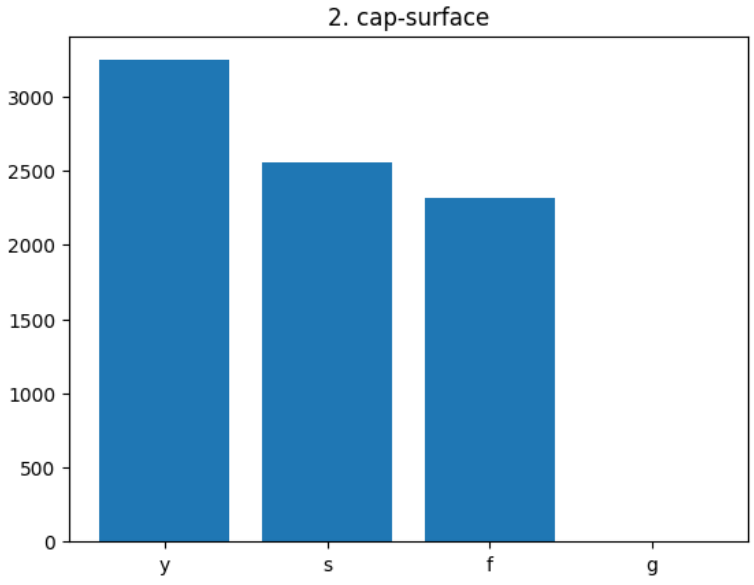
Next, we checked unique values of each feature, along with the distribution of them.

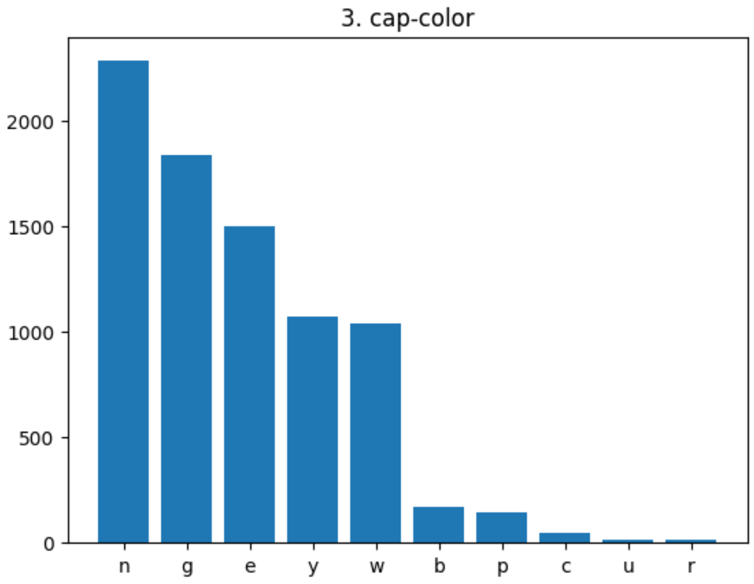
cap-shape	['x' 'b' 's' 'f' 'k' 'c']
cap-surface	['s' 'y' 'f' 'g']
cap-color	['n' 'y' 'w' 'g' 'e' 'p' 'b' 'u' 'c' 'r']
bruises	['t' 'f']
odor	['p' 'a' 'l' 'n' 'f' 'c' 'y' 's' 'm']
gill-attachment	['f' 'a']
gill-spacing	['c' 'w']
gill-size	['n' 'b']

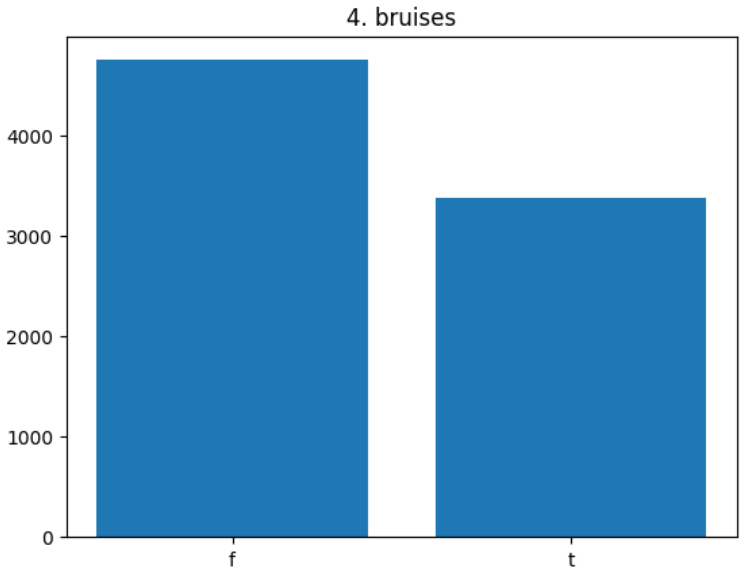
gill-color	['k' 'n' 'g' 'p' 'w' 'h' 'u' 'e' 'b' 'r' 'y' 'o']
stalk-shape	['e' 't']
stalk-root	['e' 'c' 'b' 'r' nan]
stalk-surface-above-ring	['s' 'f' 'k' 'y']
stalk-surface-below-ring	['s' 'f' 'y' 'k']
stalk-color-above-ring	['w' 'g' 'p' 'n' 'b' 'e' 'o' 'c' 'y']
stalk-color-below-ring	['w' 'p' 'g' 'b' 'n' 'e' 'y' 'o' 'c']
veil-type	['p']
veil-color	['w' 'n' 'o' 'y']
ring-number	['o' 't' 'n']
ring-type	['p' 'e' 'l' 'f' 'n']
spore-print-color	['k' 'n' 'u' 'h' 'w' 'r' 'o' 'y' 'b']
population	['s' 'n' 'a' 'v' 'y' 'c']
habitat	['u' 'g' 'm' 'd' 'p' 'w' 'l']

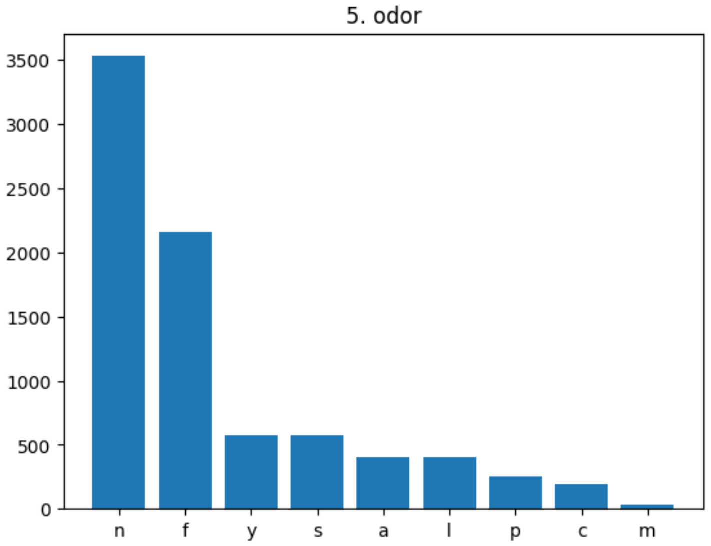
The feature ‘veil-type’ only has one unique value, so it is not plotted, and it is dropped, since it will not contribute to the classification at all.

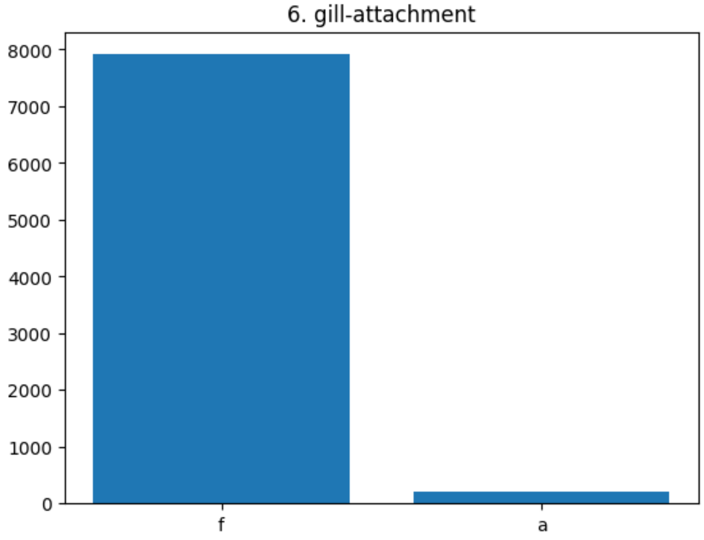


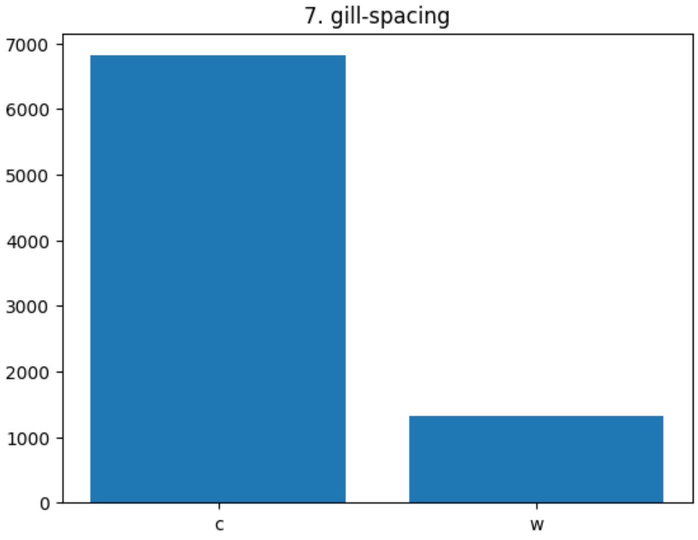


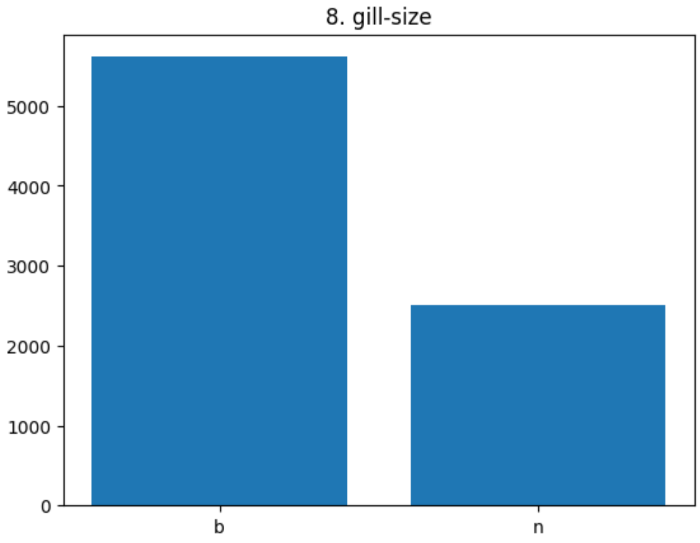

3. cap-color
0
500
1000
1500
2000
n
g
e
y
w
b
p
c
u
r


4. bruises
0
1000
2000
3000
4000
f
t

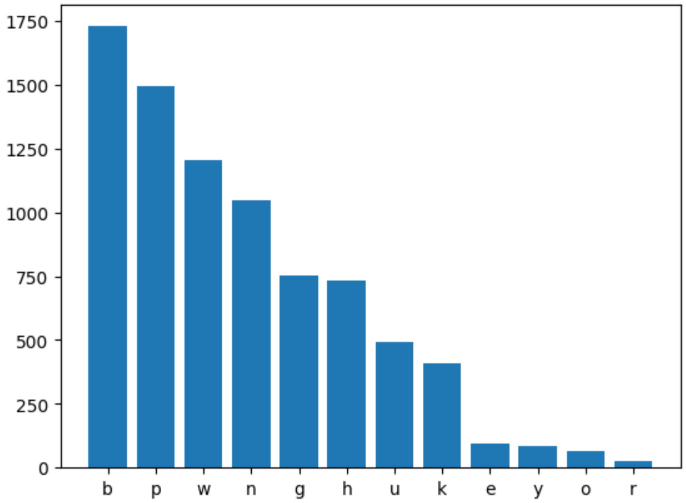

5. odor
0
500
1000
1500
2000
2500
3000
3500
n
f
y
s
a
l
p
c
m


6. gill-attachment
0
1000
2000
3000
4000
5000
6000
7000
8000
f
a

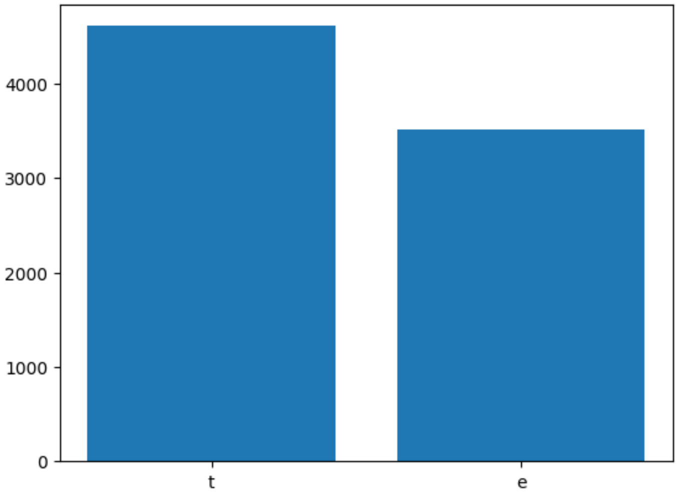

7. gill-spacing
0
1000
2000
3000
4000
5000
6000
7000
c
w


8. gill-size
0
1000
2000
3000
4000
5000
b
n

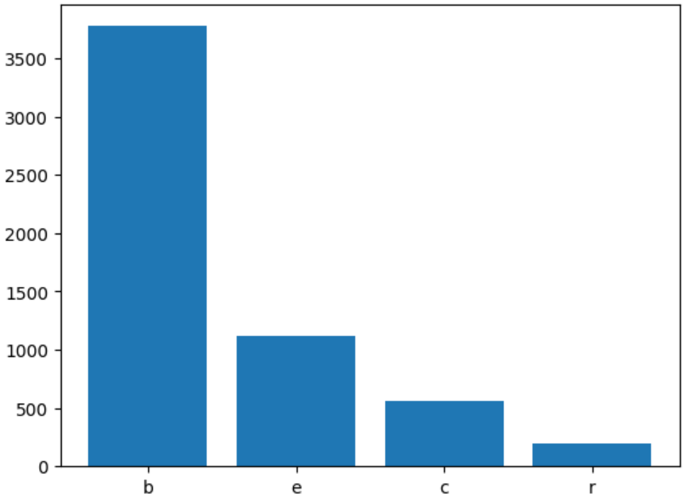
9. gill-color



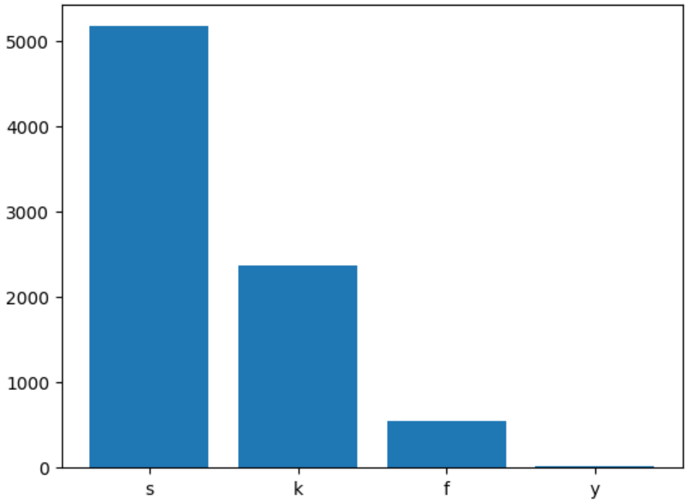
10. stalk-shape



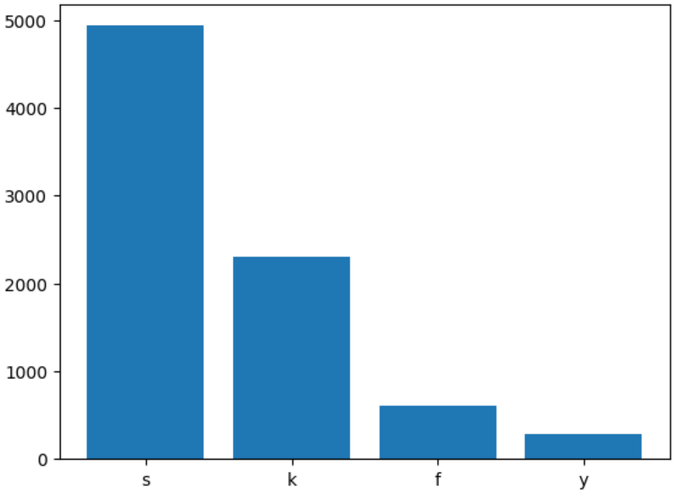
11. stalk-root



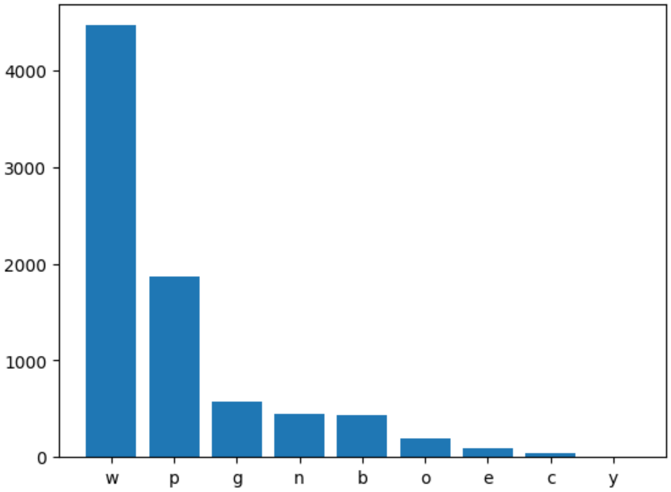
12. stalk-surface-above-ring



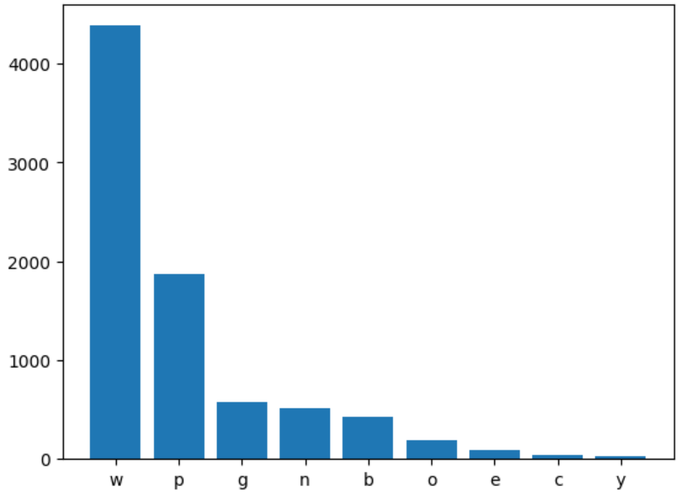
13. stalk-surface-below-ring



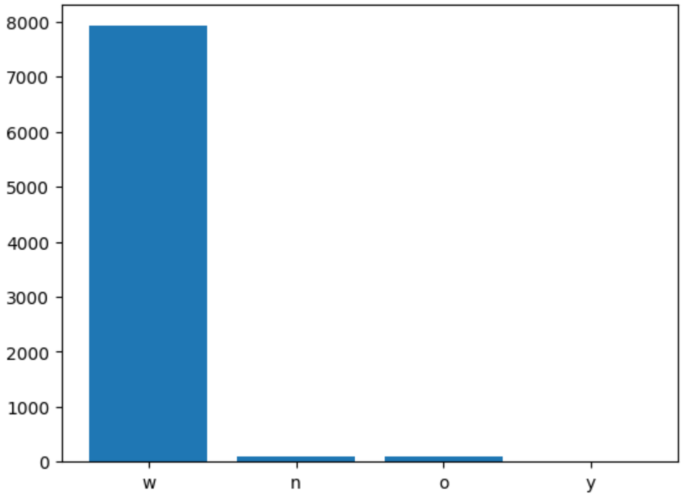
14. stalk-color-above-ring



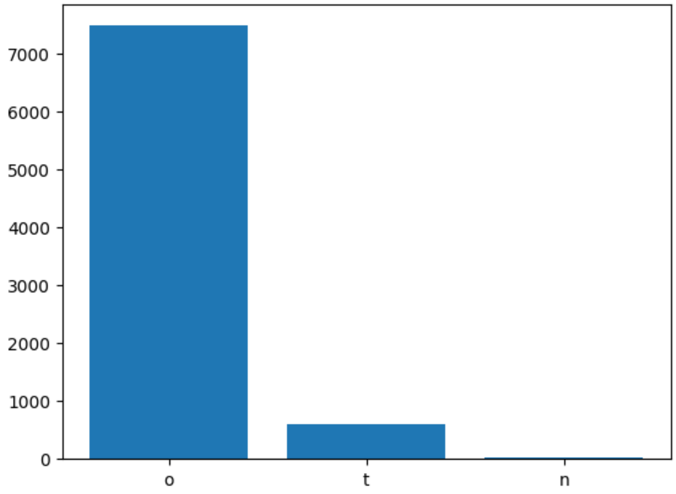
15. stalk-color-below-ring



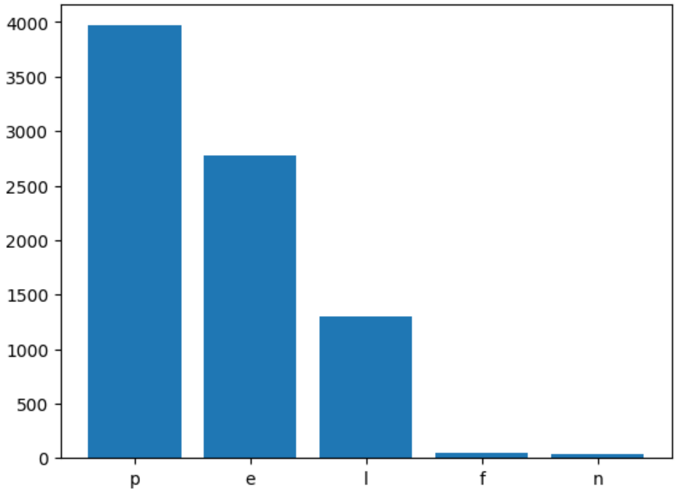
16. veil-color



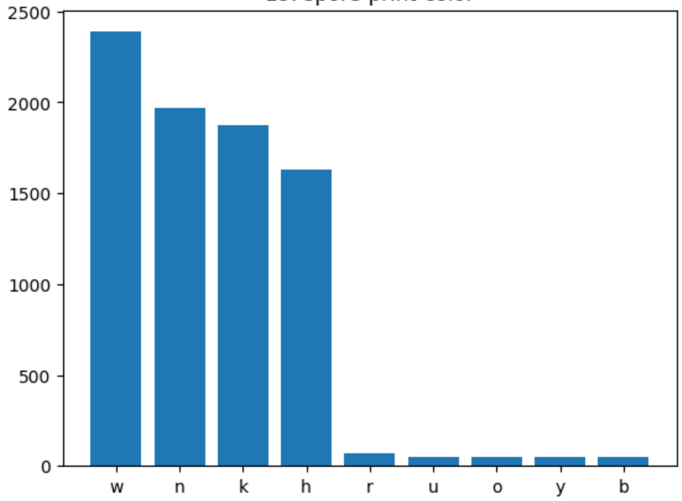
17. ring-number



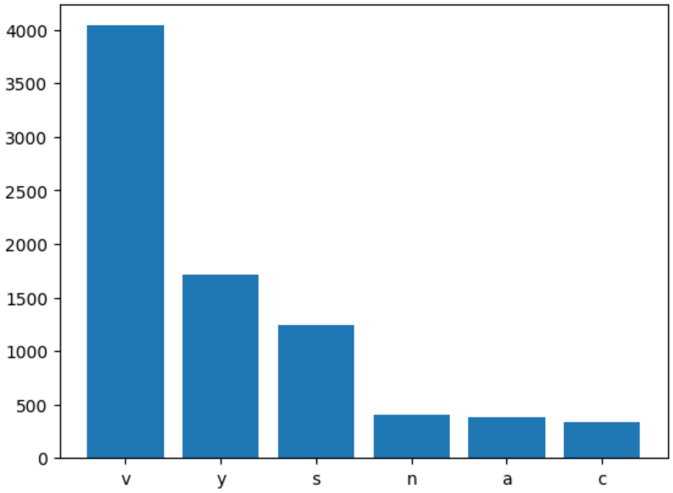
18. ring-type

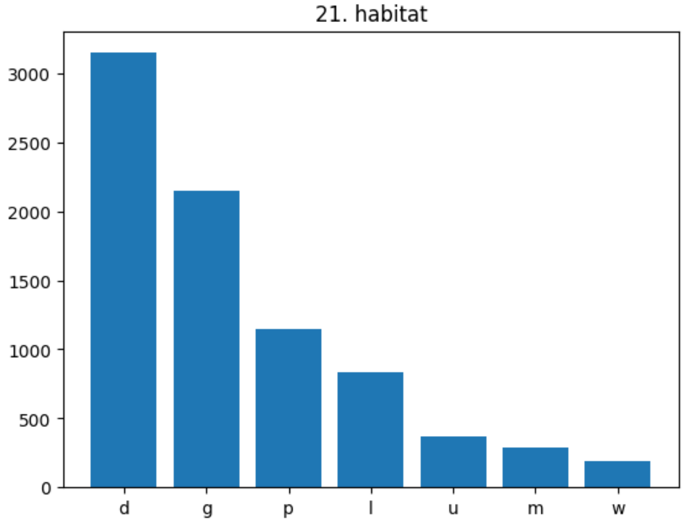


19. spore-print-color

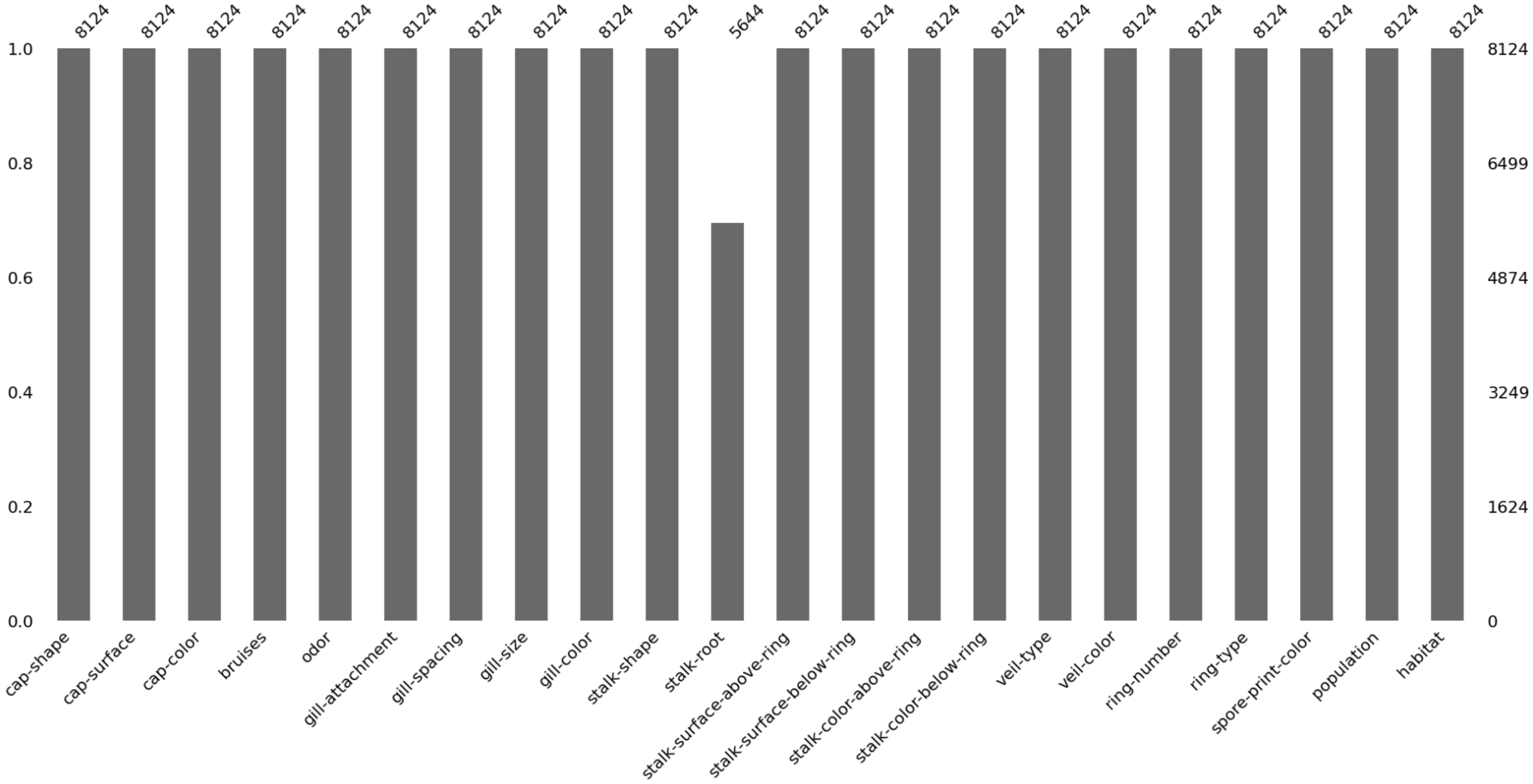


20. population





Then the dataset is searched for missing values. The plot is below.



It can be seen that no feature has missing values, except for ‘stalk-root’. 2480 samples are missing that feature. That’s around 30% of total samples. We drop the feature for ease of process.

After dropping ‘stalk-root’ and ‘veil-type’ columns, the dataset is encoded, since the entire dataset is categorical. This step is necessary even for using decision tree classifier, despite of it being particularly useful for categorical features. This is because the scikit-learn package cannot work with categorical data.

First the ordinal features are encoded manually. The features are-
'bruises', 'gill-attachment', 'gill-spacing', 'gill-size', 'stalk-shape', 'ring-number'.

For simplicity, the rest of the features are encoded via label encoding. It is risky but it is still done because the remaining 15 features will be quite big if one-hot or other related encoding is used.

2. **Model Selection and Training:** In this project, scikit-learn package is used. The models used for the classification part are-
    a. **Decision Tree-** A Decision Tree is a supervised learning algorithm. It recursively splits the data into subsets based on feature values, creating a tree-like structure where each internal node represents a feature decision and each leaf node represents a class label.
    By default, CART algorithm is used in the scikit-learn decision tree classifier. Other algorithms like ID3, C4.5 are not widely used, and they are not very optimized in python. Two variations of decision tree are used in the project, one is the default CART with Gini index, and the other is CART but with entropy as impurity measurement.
    b. **Naive Bayes-** Naive Bayes is a probabilistic classifier based on Bayes' Theorem, which assumes that the features are conditionally independent given the class label. Despite its simplicity, Naive Bayes often performs surprisingly well for classification tasks, especially with text data or datasets where features are relatively independent. It calculates the posterior probability for each class and assigns the class with the highest probability to the given instance.
    c. **Logistic Regression-** It is a linear model used for binary classification tasks. It models the probability that a given input belongs to a certain class by applying the logistic function (sigmoid) to a linear combination of the input features. Despite its name, logistic regression is a classification model, not a regression model, and is particularly well-suited for binary classification problems with well-defined decision boundaries.
    d. **K Nearest Neighbour-** K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm used for classification and regression tasks. For classification, the algorithm assigns a class label to a data point based on the majority class of its 'k' nearest neighbors in the feature space. KNN is sensitive to the choice of distance metric and the value of 'k', and works well for problems where decision boundaries are non-linear.

3**. Evaluation Metrics:** Confusion matrix is plotted for each of the models that are used. The measurement used are-

a. **Accuracy**

Accuracy is the most common evaluation metric for classification tasks. It measures the overall proportion of correctly predicted instances out of all instances in the dataset. It is calculated as:
Accuracy = (TP+TN) / (TP+TN+FP+FN)

Where:

TP = True Positives
TN = True Negatives
FP = False Positives
FN = False Negatives

Accuracy provides a general measure of the model's performance but can be misleading if the dataset is imbalanced.

b. **Precision**
Precision, also known as positive predictive value, measures the proportion of true positive predictions among all positive predictions made by the model. It tells us how many of the predicted positive instances were actually correct. It is calculated as:
Precision = TP / (TP + FP)
c. **Recall**
Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that were correctly identified by the model. It reflects how well the model captures positive instances in the data. It is calculated as:
Recall = TP / (TP + FN)
Recall is important when the cost of false negatives is high, such as in medical diagnoses or fraud detection.

d. **F1 Score**
The F1 Score is the harmonic mean of precision and recall, offering a balanced measure of a model's performance when both precision and recall are important. It is especially useful when dealing with imbalanced datasets. The formula for F1 Score is:
F1 score = 2* (Precision * Recall) / (Precision + Recall)

e. **Support**
Support refers to the number of true instances for each class in the dataset. It gives an indication of how many examples of each class exist in the dataset, and it is used to provide context when interpreting the precision, recall, and F1 score for each class.

For a given class, support is simply the number of actual occurrences of that class in the dataset. It does not affect the calculation of precision, recall, or F1 score, but it is helpful to understand the distribution of the data.

TP = True Positives

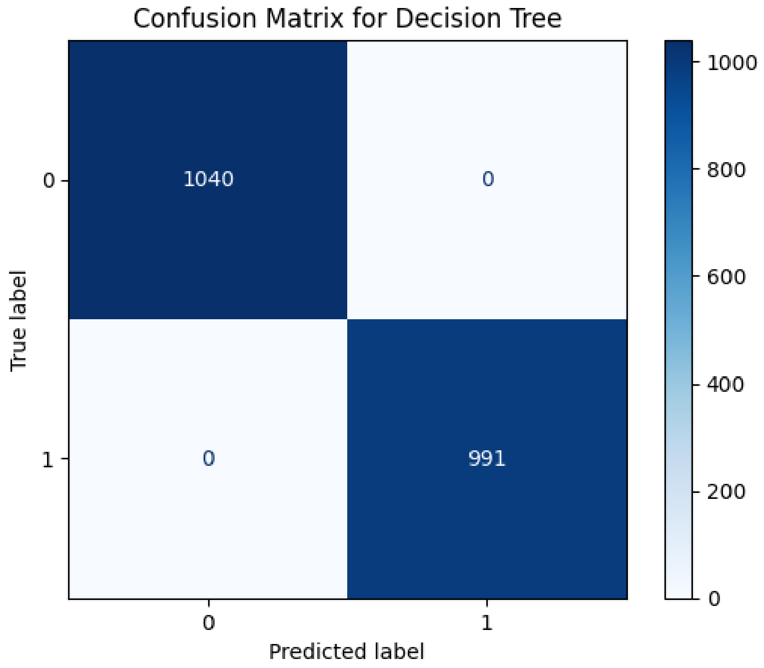
TN = True Negatives

FP = False Positives

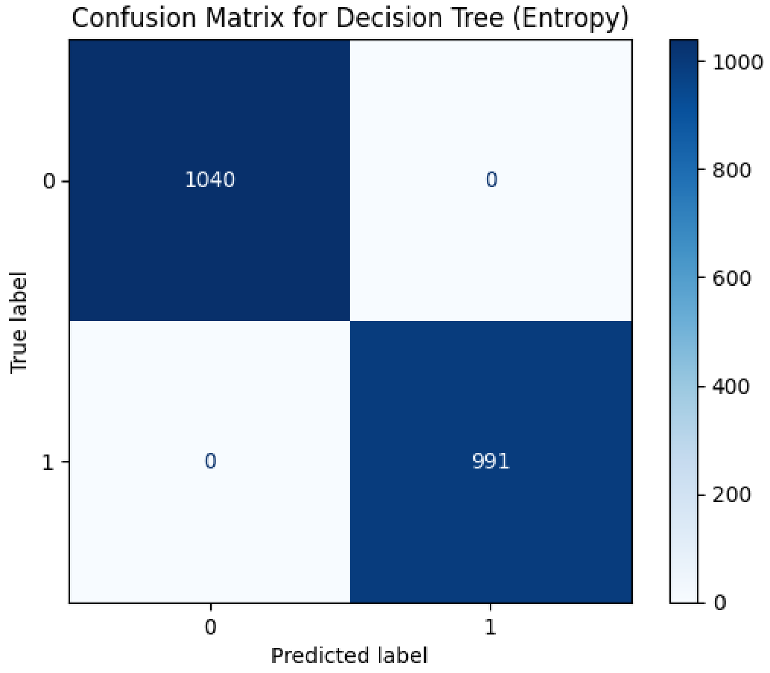
FN = False Negatives

4. **Results and Discussion:** The results of each models are given below

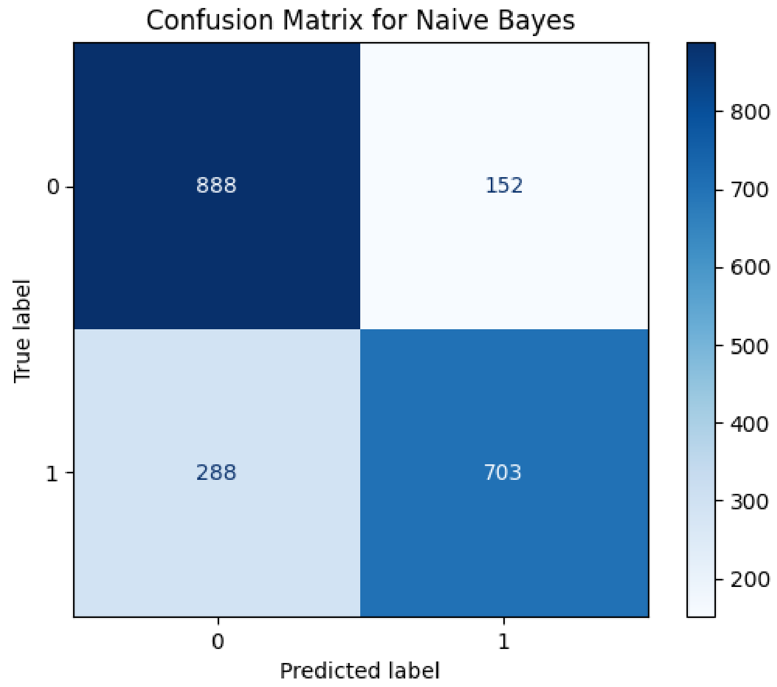
a. Decision Tree (Gini Index)-



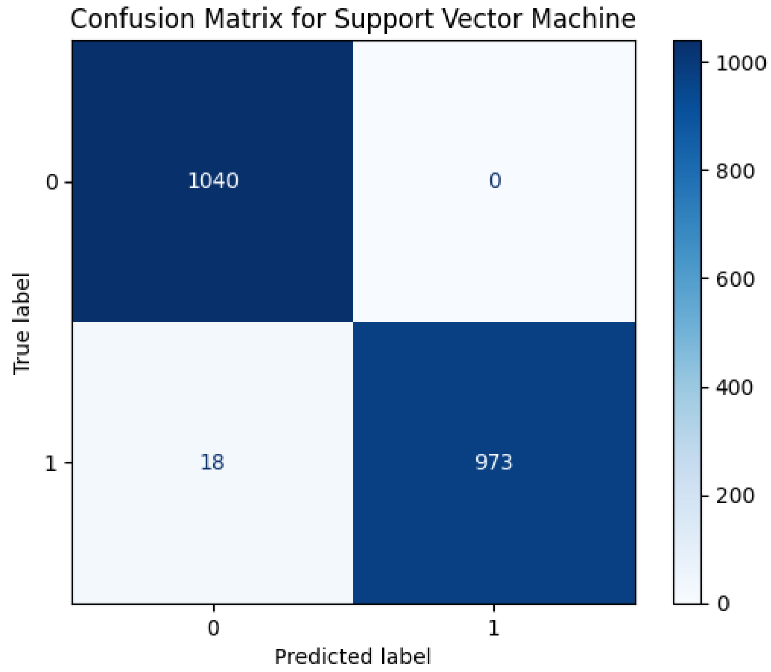
b.Decision Tree (Entropy)-



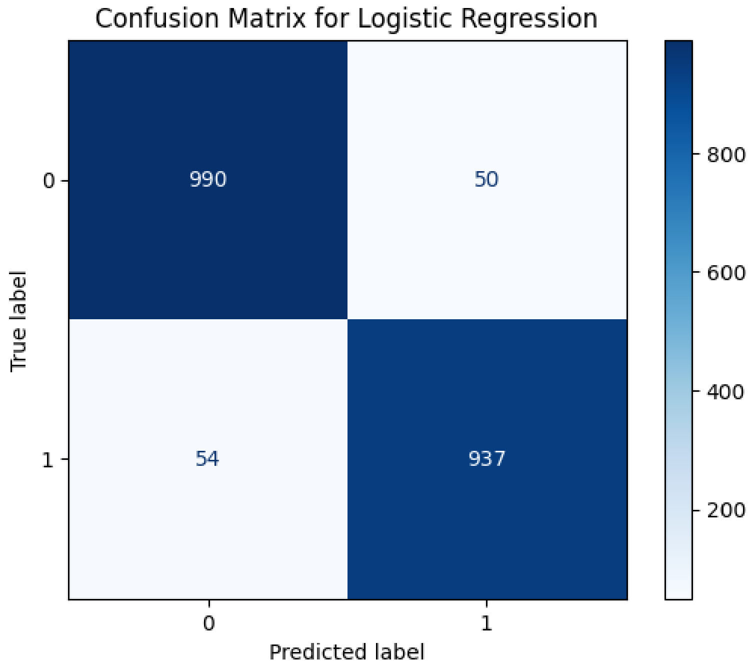
c. **Naive Bayes-**



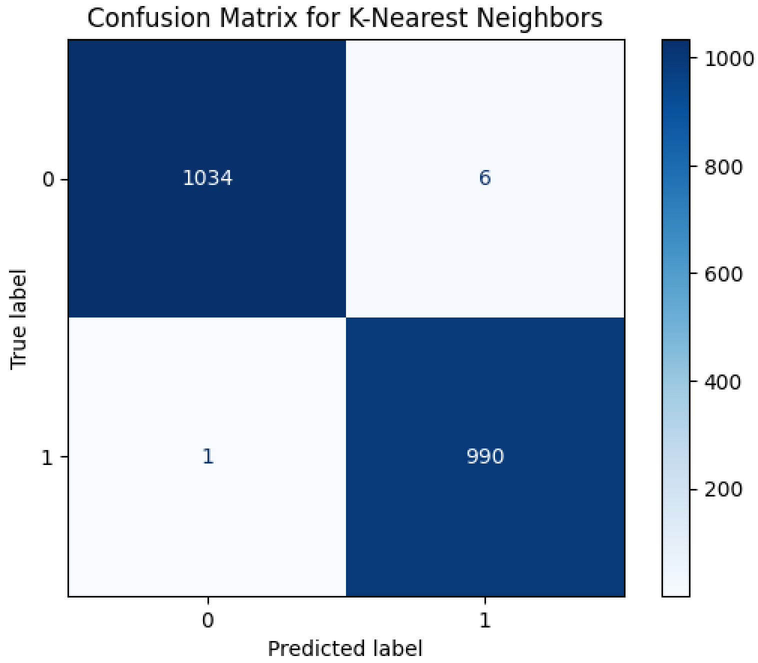
d. **Support Vector Machine-**



e. **Logistic Regression**



f. KNN-

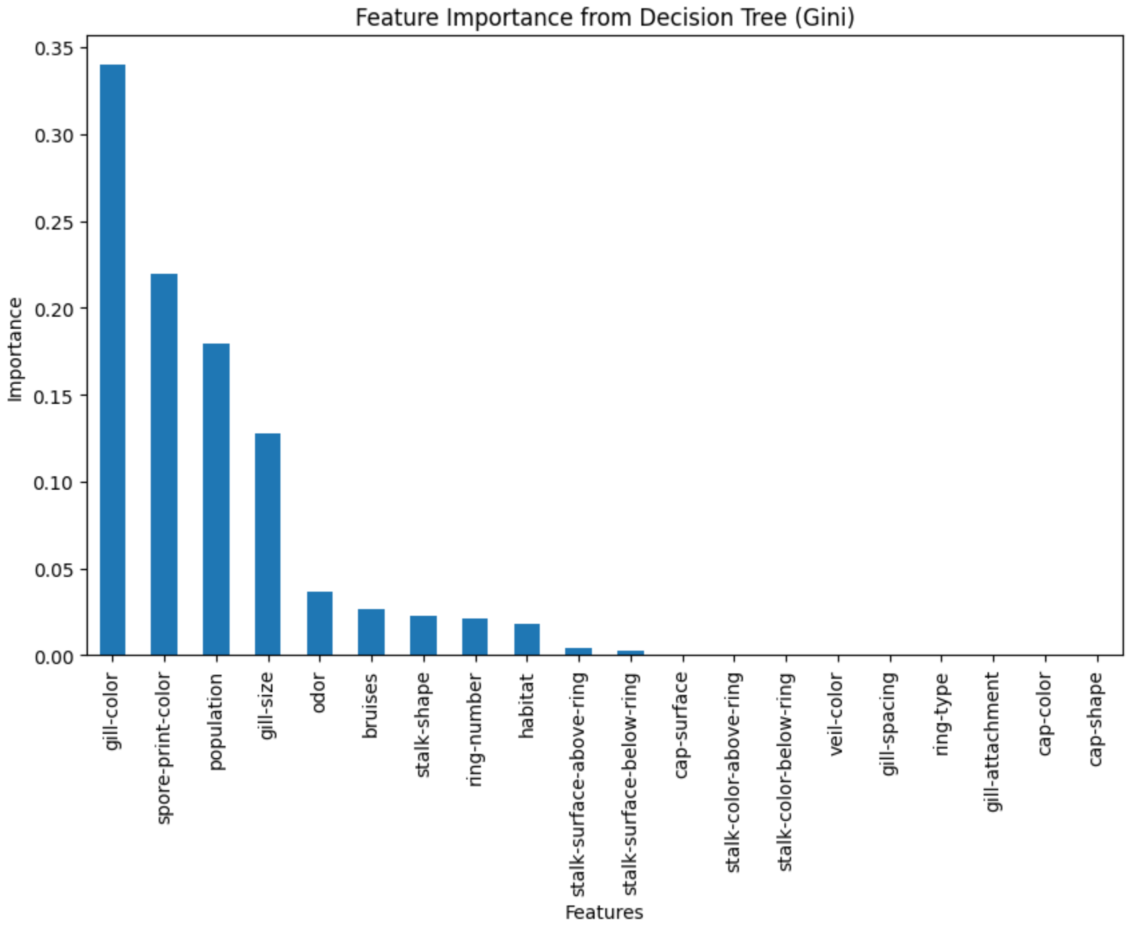


Comparison Table of Evaluation Metrics of Each Model

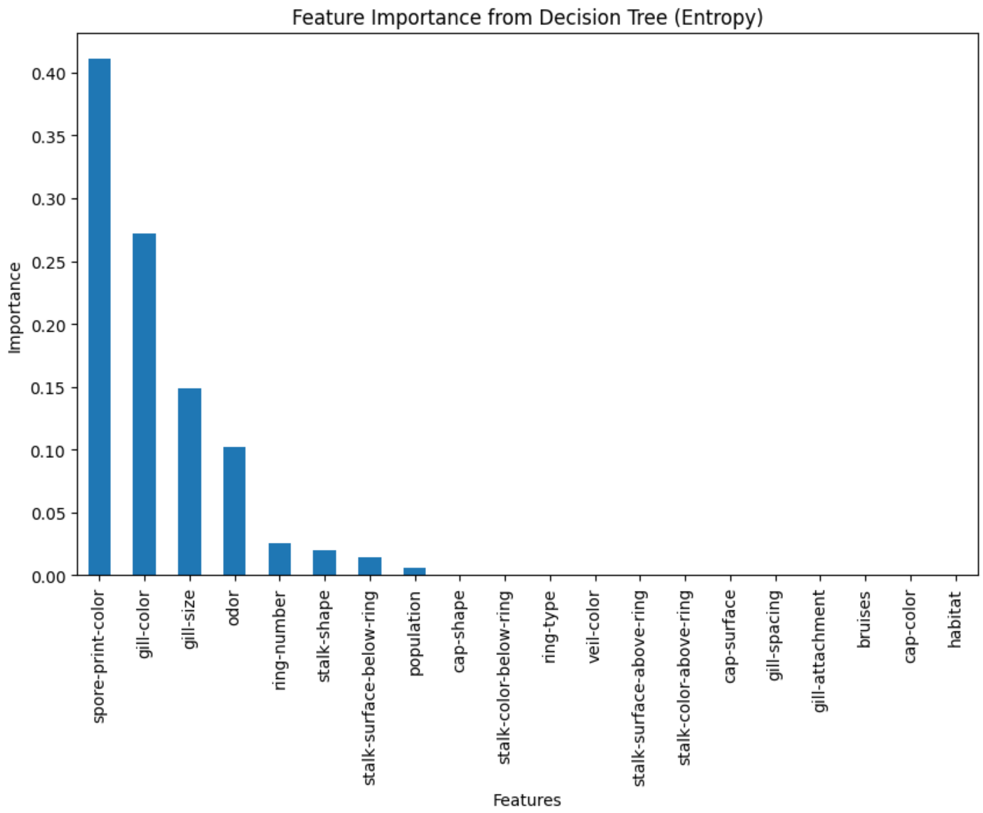
	Decision Tree (Gini)	Decision Tree (Entropy)	Naive Bayes	SVM	Logistic Regression	KNN
Accuracy	1.00	1.00	0.78	0.99	0.95	1.00
Precision	1.00	1.00	0.79	0.99	0.95	1.00
Recall	1.00	1.00	0.78	0.99	0.95	1.00
F1 Score	1.00	1.00	0.78	0.99	0.95	1.00
Support	2031	2031	2031	2031	2031	2031

Feature importance from decision tree:

a. **Gini Index-**



b. **Entropy-**



	Gini Importance	Entropy Importance
gill-color	0.339650	0.271881
spore-print-color	0.219882	0.410689
population	0.179744	0.005871
gill-size	0.128108	0.149070
odor	0.036627	0.102555
bruises	0.026889	0.000000
stalk-shape	0.022934	0.020016
ring-number	0.021189	0.025730
habitat	0.018107	0.000000
stalk-surface-above-ring	0.004260	0.000000
stalk-surface-below-ring	0.002609	0.014188
cap-surface	0.000000	0.000000
stalk-color-above-ring	0.000000	0.000000
stalk-color-below-ring	0.000000	0.000000
veil-color	0.000000	0.000000
gill-spacing	0.000000	0.000000
ring-type	0.000000	0.000000
gill-attachment	0.000000	0.000000
cap-color	0.000000	0.000000
cap-shape	0.000000	0.000000

Although the result of both decision trees are the same, the feature importance derived from them are somewhat different. But they do seem to agree on the irrelevance of some features (The last few features in above table).

It would be pretty naive to declare the features irrelevant just from two decision trees, still it's an interesting outcome.
The confusion matrix of both types of decision tree are exactly the same. The change of impurity measure did not have any effect at all. They both give 100% accuracy.

K Nearest Neighbor also works with 100% accuracy. SVM works pretty good with 99% accuracy. Logistic Regression also seems to work fine, but it takes some time to converge. Maximum number of iterations was changed to 10,000 for it to stop training automatically.
Naive Bayes gives an accuracy of 78%, which is not a good result, but it's probably because Naive Bayes algorithm assumes all the features are independent of each other, which is certainly not true in most dataset.
The different models give average to excellent accuracy without much preprocessing. This indicates that the quality of the dataset is extremely good. No feature selection or dimensionality reduction is done in the preprocessing step, except for dropping a feature with null values, and one with only one value.

Limitation and Future Aspect:

1. We used only 6 classifiers. Many other classifiers like XGBoost, Adaboost, Artificial Neural Network could be used for the classification along with already used, to better compare the result. However the limited time is not enough for all the classifiers.
2. There are various algorithms for Decision Tree Classifiers, but currently in Python, only CART algorithm is used efficiently in the scikit-learn module. Other algorithms like ID3, C4.5 are either not available or not efficient enough in other modules. In future languages like R can be used for better results.
3. Although decision trees are particularly useful for categorical data. The scikit-learn module requires input data to be encoded in numerical form. Till now there is no scope to directly use categorical data. In future when support for categorical data comes into scikit-learn, it might give more information about feature importance. Or other languages like R can be used.
4. For simplicity and lack of time, the categorical data is encoded by label encoding, which might contribute to some error in the result. And 'stalk-root' feature is dropped, but in future we can remove the samples with missing values from the dataset, and do the whole classification again, to see if it is actually an important feature or not.
5. X-AI (Explainable AI) can be used to extract feature importance from various classifiers, given more time.

   Overall, the project is completed within a tight schedule. But with more time, most of the limitations can be overcome and better conclusions can be drawn.

Conclusion:
In conclusion, this project uses several classification techniques to classify poisonous and edible mushrooms from a dataset of all categorical features. The dataset Mushroom (https://doi.org/10.24432/C5959T) is a high-quality dataset, which includes null values of only one feature. Without much preprocessing, a good accuracy can be reached. KNN, Decision Tree, Support Vector Machine, Logistic Regression works pretty well. Naive Bayes falters (78% accuracy) probably because there is underlying dependency between features, but it cannot be said for sure, from this small study. Further study is needed in order to reach a solid conclusion. The research papers used in literature review, use feature extraction and get to almost similar accuracy. Our project uses 20 features to classify. So the complexity is quite high. Simple approaches should be preferred if it is only used for classification. However feature extraction will be useful if done. It can be done in the future, with more time.

Bibliography:

1. **Classification and selection of the mainfeatures for the identification of toxicity in Agaricus and Lepiota with machine learning algorithms**, *Jacqueline S. Ortiz-Letechipia, Carlos E. Galvan-Tejada1, Jorge I. Galván-Tejada, Manuel A. Soto-Murillo, Erika Acosta-Cruz, Hamurabi Gamboa-Rosales, José María Celaya Padilla and Huizilopoztli Luna-García*
2. **IoT enabled mushroom farm automation with Machine Learning to classify toxic mushrooms in Bangladesh**, *Hasibur Rahman, Md. Omar Faruq, Talha Bin Abdul Hai , Wahidur Rahman, Muhammad Minoar Hossain, Mahbubul Hasan, Shafiqul Islam, Md. Moinuddin , Md. Tarequl Islam, Mir Mohammad Azad*
3. **Scikit-learn: Machine Learning in Python**, *Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.*
4. **Mushroom Dataset, DOI 10.24432/C5959T**, *UCI Machine Learning Repository*