Binary Classification of Edible and Poisonous Mushrooms using Classical ML techniques

SUBMITTED BY:
Priyanka Borwanker(BT19CSE018)

UNDER THE GUIDANCE OF:
Dr. Ravindra Keskar

Department of Computer Science and Engineering, VNIT Nagpur

1. **Problem Statement**: Using Machine Learning to determine whether a mushroom is edible to eat or poisonous based on certain features.

2.

Input Parameters(23)	**Output Parameters(1)**
cap-shape cap-surface cap-color bruises odor gill-attachment gill-spacing gill-color stalk-shape stalk-root stalk-surface-above-ring stalk-color-above-ring stalk-color-below-ring veil-type veil-color ring-number ring-type spore-print-color population habitat	class (edible/poisonous)

3. **Dataset Used:** University of California, Irvine - Machine Learning - Mushroom Classification | Kaggle
    a. **Insights about data:**
        i. Dataset contains 8124 data samples with 22 features each, classified into edible and poisonous.
        ii. Each data sample consists of class type and feature set (nominal values).

	count	unique	top	freq
class	8124	2	e	4208
cap-shape	8124	6	x	3656
cap-surface	8124	4	y	3244
cap-color	8124	10	n	2284
bruises	8124	2	f	4748
odor	8124	9	n	3528
gill-attachment	8124	2	f	7914
gill-spacing	8124	2	c	6812
gill-size	8124	2	b	5612
gill-color	8124	12	b	1728
stalk-shape	8124	2	t	4608
stalk-root	8124	5	b	3776
stalk-surface-above-ring	8124	4	s	5176
stalk-surface-below-ring	8124	4	s	4936
stalk-color-above-ring	8124	9	w	4464
stalk-color-below-ring	8124	9	w	4384
veil-type	8124	1	p	8124
veil-color	8124	4	w	7924
ring-number	8124	3	o	7488
ring-type	8124	5	p	3968
spore-print-color	8124	9	w	2388
population	8124	6	v	4040
habitat	8124	7	d	3148

4. **Proposed Solution:**
    a. **Data Preprocessing:**
        i. Remove any missing or null attributes from the dataset.
        ii. Dimensionality Reduction: Remove attributes with low variance for e.g. veil-type has only one value.
        iii. Encode labels to digits
    b. **Possible Algorithms for Classification:**
        i. **SVMs:**
            1. Since this is a problem of binary classification, SVMs are an obvious choice.
            2. It is not sure whether the data is linearly separable or not, so we can try SVMs with different types of kernels (linear, rbf, etc)
        ii. **Decision Tree Learning:**
            1. Under the assumption that all the attributes are independent of each other, we can apply decision tree learning.
            2. Simple model which can use the whole dataset and all of the features
            3. Highly intuitive solution which can be easily visualized.
        iii. **Multi Layer Perceptrons:**
            1. Multilayer perceptrons are a good option for supervised learning when the relationship between the attributes and output classes is not apparent.
            2. A multilayer perceptron can model even non linear classification problems quite well.
    c. **Libraries Used:**
        i. **Pandas:**
            1. Open source, easy to use, data management framework
        ii. **Scikit-learn**
            1. Simple and efficient tools for predictive data analysis
            2. Built on NumPy, SciPy, and matplotlib
            3. Open source, commercially usable
    d. **Implementation:**
    https://colab.research.google.com/drive/1I5a6Dsk1G0Nop8ZdWbczVkWX4Q0pt3bX?usp=sharing
        i. SVMs implemented with two types of kernels (rbf and linear)
            1. Linear - standard linear svm
            2. Rbf - radial basis function, can model non-linearity in data as well
        ii. MLP Classifier:
            1. ReLu activation function
            2. 1 hidden layer of 70 neurons(decreased to avoid overfitting)
            3. sgd optimizer
            4. Learning_rate = 0.0005 (decreased to avoid overfitting)
        iii. Decision Tree:
            1. Pre Pruning of decision trees with maximum depth set to 4

Before Pruning:

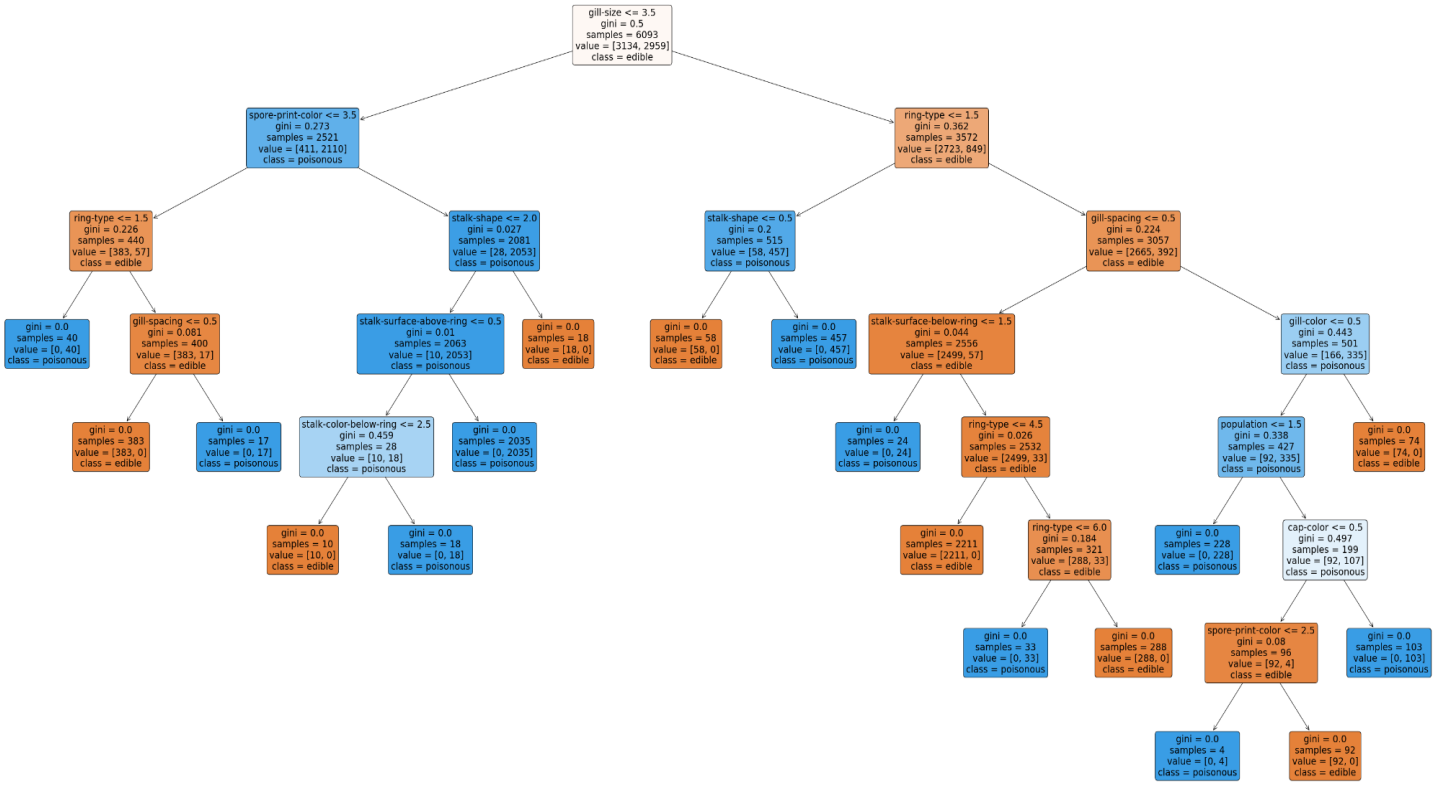


Fig: Highly irregular and too specific decision tree

After Pruning:

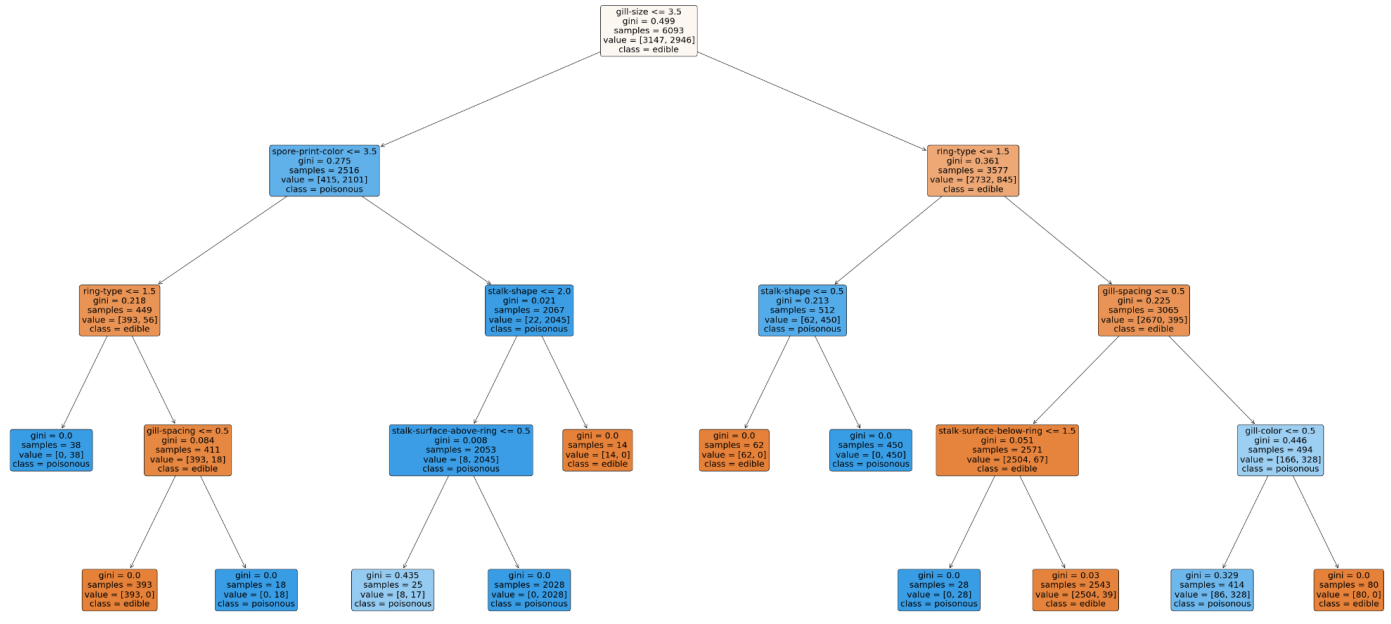


Fig: Balanced decision tree

e. **Accuracy Measure:**

i. We track when the predicted class and actual class is the same and include that in our score, else the score does not increase.

ii. **K-Fold Cross Validation**: The dataset is divided into k parts and then k-1 parts are used for training and the kth part is used for validation. The model accuracy is then defined as the average of the validation accuracy of each part.

iii. **Results:**

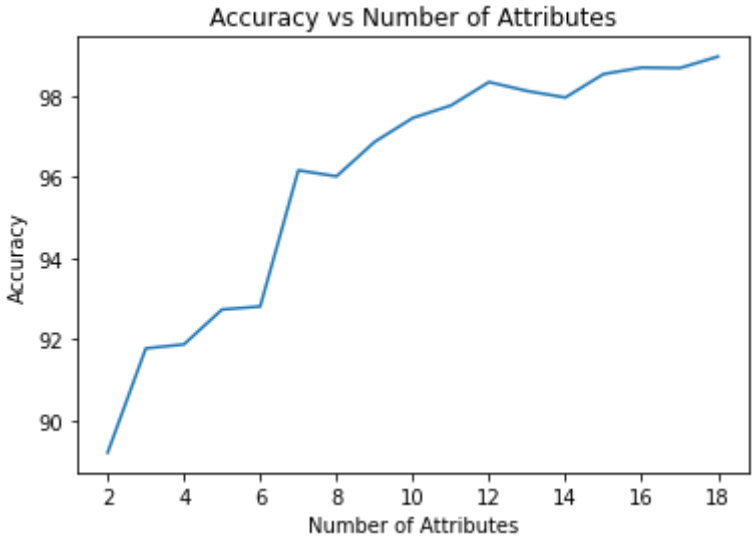
	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Average Acc.**
SVM (linear kernel)	95%	92%	94%	95%	94.04%
SVM (rbf kernel)	99%	99%	99%	99%	98.93%
MLP Classifier	97%	96%	98%	96%	96.80%
DecisionTrees	98%	98%	97%	98%	97.78%

➔ *SVMs with the rbf kernel are giving the best results for binary classification, which is in accordance with the fact that SVMs are still considered the gold standard for binary classification.*

f. **Feature Selection: Can we achieve similar accuracy with less features?**

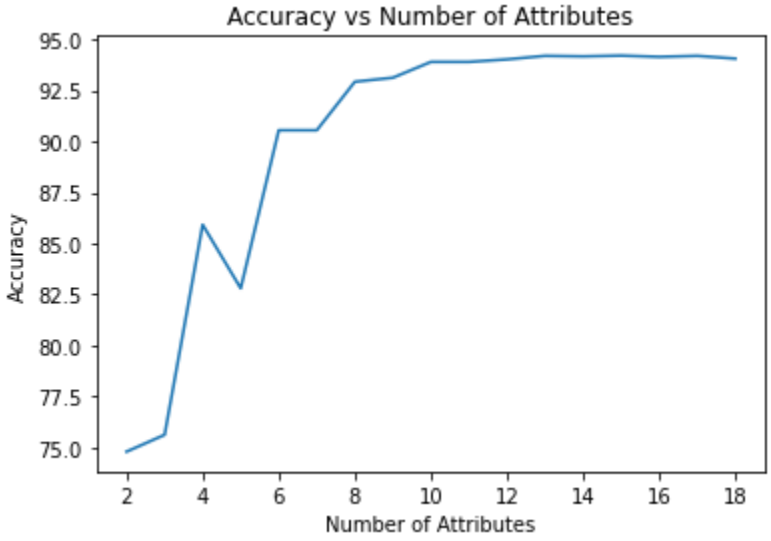
i. **Chi-Square Analysis -** Using chi-square analysis we can pick the top most independent attributes and only use them for classification and check the accuracy. Here are some plots which show the variation of accuracy when we chose 2-18 top features.

1. SVC (kernel = 'rbf')



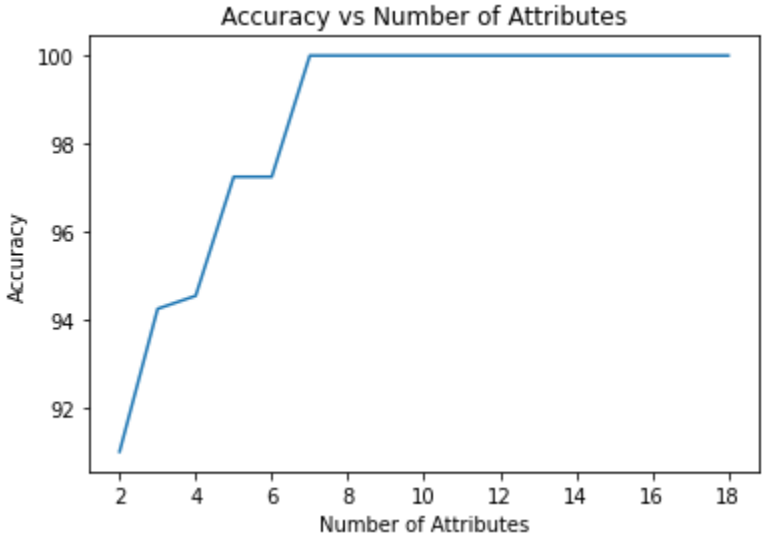
Here we can see that accuracy is increasing almost linearly with the number of features.

2. SVC (kernel = ‘linear’)



Here we can observe that accuracy is approximately constant after 10 topmost features.

3. Decision Trees



In this, accuracy is the same whether we use the topmost 7 features or all the 18 features. So it is better to use only top 7 features as others are just increasing the computational complexity.

g. Advantages and Disadvantages

	Advantages	**Disadvantages**
SVMs	● Computationally inexpensive ● More productive in high dimensional spaces ● Better when data is linearly separable with a clear margin	● Does not work well when relationship between attributes may not be linear (Our case) ● Does not work well in cases of noise or large dataset
MLP	● Can handle non-linearity in data well ● Quick predictions ● Can handle large amount of data easily	● Computationally more expensive than DecisionTrees and SVMs
DecisionTrees	● Less preprocessing of data is required ● Normalization/scaling not required ● Missing values has minimal impact on decision trees	● Noise can lead to unbalanced decision trees ● Training time is expensive but testing is fast ● Overfitting