Classifying Mushrooms from Agaricus and Lepiota Family

Data Set: Mushroom

Vishal Shashikant Patil
vspatil@usc.edu

Venkata Meghana Achanta
vachanta@usc.edu

May 7, 2023

1 Abstract

Agaricus and Lepiota are two genus of mushrooms which contain a mix of common edible mushrooms and toxic mushrooms. In this problem we try to develop a machine learning algorithm which can classify these mushrooms with sufficiently high accuracy. The dataset used is the UCI mushroom dataset [1]. The categorical variables were transformed into 92 one hot encoded variables. Various feature engineering methods like combining features, Principal Component Analysis (PCA), Univariate Feature Selection (UFS) and Recursive Feature Elimination (RFE) were used to reduce feature dimensions. The method which gave the highest accuracy with greatest reduction in the number of features was adopted for model selection. A simple logistic regression model was used to compare accuracies of different feature reduction methods. 46 features obtained after the UFS method were finalized the train and test features. Logistic regression, Support Vector Machine (SVM) with RBF kernel, Perceptron, K Neighbours, Random Forest and Multilayer Perceptron (ANN) models were tested. We found that random forest classifier with 100 decision trees provided the best accuracy of 99.53% and an F1 score of 99.464%.

2 Introduction

2.1 Problem Assessment and Goals

UCI Mushroom Data set [1] is used in this project. The dataset consists of 61069 mushrooms from 173 species of mushrooms (353 mushrooms per species) from Agaricus and Lepiota Families. It has 16 columns, 15 features and a "class" column with labels as 'e' (edible) and 'p' poisonous for each mushroom which makes this a binary classification problem.

Following are the main goals that we have aimed to achieve in this project:

- Classify mushrooms as poisonous or edible with high accuracy (above 90%)
- Select a model with good F1 score (low false positives and false negatives)
- Explore and perform extensive feature engineering methods to identify the best features for classification
- Perform hyperparameter search for the models to be tested through cross validation to examine each model's performance carefully.
- To identify the most common categorical features of poisonous mushrooms.
- Propose the best model with low computational complexity, high accuracy and F1 score and high scalability.

3 Approach and Implementation

The process followed in the project has been summarized in the flowchart as shown below. We perform initial exploration of the data and one hot encode all the categorical features in the preprocessing stage, then apply 4 methods namely PCA, combining features, UFS and RFE for feature selection and dimensionality reduction in Feature Selection stage. We then set a trivial system and a nearest means classifier models as baseline against which performance of other models will be compared in setting baseline models stage. We then test 6 different classifiers, perform hyperparameter search for each model and evaluate performance in the Model selection, Hyperparameter search and Performance evaluation stages.

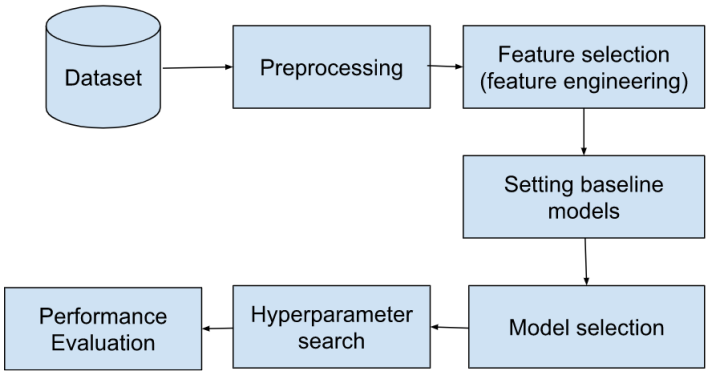

Figure 1: Implementation Flowchart

3.1 Dataset Usage

The table summarizes the usage of train and test sets:

Dataset	Number of data points	Number of Feature	Features after Preprocessing	Reduced Features
Training set	42748	15	92	46
Test set	18321	15	92	46

Table 1: Dataset Distribution and Features

Cross validation was used in hyperparameter search while training different models, the table below summarizes train set usage for cross validation.

Total number of data points from train set	42748
Number of features during cross validation	46
Number of runs	1
Number of Folds	5
Number of data points in train set during each fold of cross validation	34196
Number of data points in each validation set	8549

Table 2: Train Set Usage for Cross Validation

As shown in the tables above, the total data set was split into train and test sets. Originally the dataset had 15 categorical features. After feature engineering and dimensionality reduction a total of 92 new features were added from which 46 best features were selected.

After finalizing the 46 best features, we extract the train test sets with these 46 features from the original dataset with 92 features which are then used for model selection.

During model selection 5 fold cross validation is used for each model of the 6 models in order to perform hyperparameter search. The test set is not used at this stage. The train set is split into 5 groups with each split containing about 8549 data points. During each fold one of the splits is used as a validation set and the other 4 splits are used as training sets.

3.2 Preprocessing

The preprocessing step involved converting the categorical columns to one hot encoded feature. The table below shows the number of unique values for each of the categorical features and the total number of features after one hot encoding. Out of the 15 features there are 3 numerical features and 12 categorical features. Only the 12 categorical features were considered for one hot encoding.

Feature	Number of Unique Values
Cap Shape	7
Cap Surface	11
Cap Color	12
Does bruise or bleed	2
Gill Attachment	7
Gill Spacing	3
Gill Color	12
Stem Color	13
Has Ring	2
Ring Type	8
Habitat	8
Season	4
Numerical Features	3
Total(After One Hot Encoding)	92

Table 3: Categorical Features of the Dataset

3.3 Feature Engineering and Dimensionality adjustment

From a high level, in this stage we tried to add new features instead of one hot encoding and evaluated logistic model regression based on the training set with new features added. We compared this with logistic regression model performance on the train set with one hot encoded features and then decided which approach would be better manually adding new features or using one hot encoded features.

3.3.1 Adding new features

The process of adding new features is as follows:

1. Group the data based on one of the categorical features and one of the numerical features
2. Calculate a statistic namely mean, max, min or median based of the numerical features of the grouped data
3. Add this as a new feature instead of one hot encoding the categorical feature

An example of the above process is given below:

- Group the data based on cap-color (categorical feature) and cap-diameter (numerical feature).
- There are 12 different cap-colors hence there will be 12 groups.
- Calculate means from each of the 12 groups.
- Create 12 new features where each feature each represent one of the cap colors.
- For every column insert the calculated mean value to only those rows where the cap-color matches the cap-color indicated by the column. For example, if the first column corresponds to yellow cap color then it will have 0's in all rows where the cap-color column is not yellow and the mean value where the cap-color column is yellow.

This process is repeated for each categorical feature. The table below shows the total number of possible combinations using this method.

Grouping	Values	Count
Categorical features	Cap-shape, cap-surface, cap-color, does-bruise-or-bleed, gill-attachment, gill-spacing, gill-color, stem-color, has-ring, ring-type, habitat, season	12
Numerical features	cap-diameter, stem-height, stem-width	3
Statistic	Average, Minimum, Maximum, Median	4
Total number of possible grouping combinations		144

Table 4: Categorical Combinations of Features

The table below gives the top 5 combinations based on logistic regression model accuracy trained on the train set with these new features.

Combination	Accuracy (Logistic Regression classifier)
gill-attachment, stem-width, max	63.577%
gill-attachment, stem-height, median	63.539%
gill-attachment, stem-height, max	63.501%
gill-attachment, cap-diameter, mean	63.490%
gill-attachment, cap-diameter, mean	63.490%
All features after one hot encoding	77.616%

Table 5: Combination of Features

From the table above we can see that grouping of features to create new features is not providing better results than using one hot encoding. Grouping based on 2 categorical features and one numerical feature could be tested but was not covered in this scope.
Based on the results, the decision to use one hot encoded features was made.

3.3.2 Feature Selection

After finalizing to move ahead with one hot encoded features, we had 92 features. Various approaches with Univariate Feature Selection and Recursive feature elimination were tried in order to reduce the feature dimensions. These approaches are discussed in this section. For this we had referred to the Python Notebook and Notes provided in Discussion 13.[2]

- **Univariate Feature Selection (UFS)**
  From the set of 92 features, we determined that selecting the best 46 features and then training the logistic regression model on the dataset with these selected features resulted in almost the

same accuracy as training the model with all 92 features. Since most of the features are categorical in nature, chi squared statistical test was performed to assess the goodness of fit between each categorical variable and the 'Class' column. The number 46 was determined in through the following process:

1. Iteratively select an increasing number of k best features depending on the their chi squared significance
2. For k best features selected train a logistic regression model
3. Measure the accuracy on the test data and store the results
4. Get the number of features that resulted in highest accuracy

The following figure shows a plot of accuracy vs number of k best features:

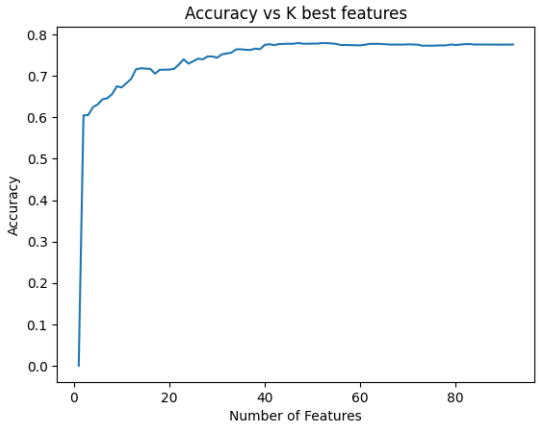

Figure 2: Accuracy vs k best features

From this plot we determined that for after k = 46 the gain in performance in terms of model accuracy plateaus. Therefore, the decision to select the top 46 features was made. The selected features are tabulated below:

- **Recursive Feature Elimination (RFE)**

  RFE recursively selects smaller subsets of features based on an estimator. The estimator used was a logistic regression model. The role of estimator is to assign weights to the features. For each iteration the top ranking features obtained by the coef_attribute are retained and other are dropped. This method selected 51 best features from the total 92 features.

- **RFE on features selected by UFS**
  In this approach we tried to apply RFE of the 46 features selected by UFS to check if further reduction in the number of features was possible. It was observed that all the 46 features selected by UFS were being considered by RFE and there was no further reduction in dimensionality. Therefore we did not continue with this approach.

- **Features selected by both UFS and RFE**
  In this approach we tried to select the features that were commonly selected by UFS and RFE. The result was there were 32 features common to UFS and RFE. On training a logistic regression model with these 32 features we saw that the model accuracy was 75.804%. Therefore we decided to discontinue with this approach and finalize the features selected by UFS.

  The table below summarizes the analysis done:

gill-attachment', 'e'	ring-type', 'm'	stem-color', 'p'	cap-surface', 'h'	cap-shape', 'c'	stem-color', 'n
cap-color', 'r'	cap-color', 'e'	gill-color', 'r'	cap-surface', 'g'	gill-color', 'k'	gill-attachment', 'p'
gill-spacing', 'd'	cap-shape', 'o'	cap-surface', 'y'	does-bruise-or-bleed', 'a'	season', 's'	habitat', 'h'
stem-color', 'b'	cap-color', 'k'	cap-surface', 's'	habitat', 'u'	ring-type', 'l'	stem-color', 'u'
gill-spacing', 'n'	habitat', 'l'	cap-color', 'o'	stem-color', 'y'	habitat', 'w'	stem-color', 'r'
cap-shape', 'f'	cap-color', 'l'	gill-color', 'w'	stem-color', 'o'	cap-surface', 'l'	season', 'u'
gill-color', 'w'	cap-surface', 'd'	ring-type', 'z'	cap-shape', 'p'	habitat', 'p'	stem-color', 'f'
stem-color', 'l'	stem-height	stem-width	cap-diameter		

Table 6: 46 Features Selected

Approach	**Reduction in Dimensions**	**Accuracy with reduced dimensions**
UFS	From 92 to 46	77.971%
RFE	From 92 to 51	77.638%
RFE on UFS	From 92 to 46(Same as UFS)	77.971%
Common to RFE and UFS	From 92 to 32(Common Features)	75.804%

Table 7: Feature Selection

We selected the 46 features given by the UFS approach as it gave the lowest number of features with highest accuracy and the approach was computationally least expensive among all approaches.

3.4 Cross Validation

Cross Validation is a technique used to evaluate the performance of a model on an independent dataset. In cross validation, the dataset is run through a specified number of folds. For our dataset, the number of folds used was 5 as it provided the best accuracy.

The dataset is divided into 5 subsets where each subset was subsequently divided into a training set and a validation set. The model was fitted over the training set and it was then tested on the validation set. The results from each fold are then averaged to give an overall estimate of the model's performance.

Cross-validation has the advantage of providing a more accurate estimate of the model's performance than merely training and testing on a single dataset. Cross-validation decreases the risk of overfitting to a specific dataset by utilizing multiple folds and provides a better approximation of how the model will perform on new, unseen data.

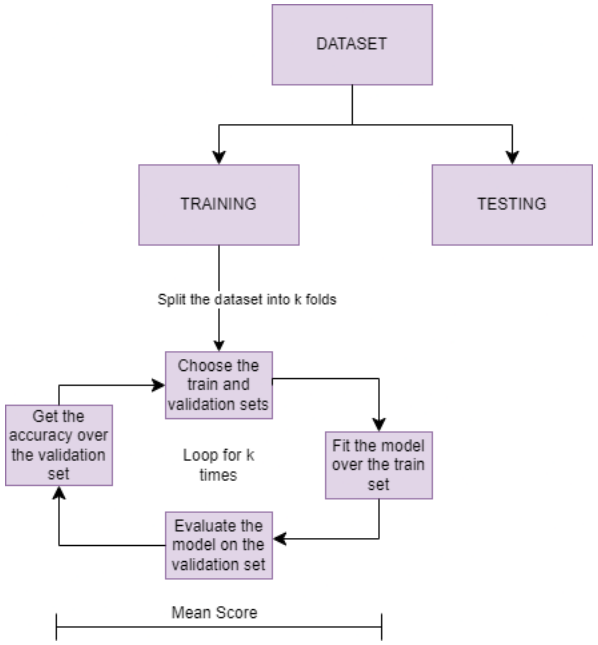

Figure 3: Cross Validation

While performing cross validation, each subset had about 8550 data points, where the training set had about 6840 data points and the validation set had 1710 data points.
The experiments were run using cross-validation and the test set was used to predict the accuracy of the model depending on each hyperparameter.

3.5 Training, Classification or Regression, and Model Selection

For our dataset, UFS had provided the best accuracy with 46 features. We used UFS as a feature selection method and used it to obtain the final dataset. We then used cross-validation to evaluate the performance of various models. After training each model, we calculate the best accuracy and the F1 score to decide which model performed the best.

3.5.1 Trivial System

In the trivial system, the model follows a distribution such that it assigns the class to each data point randomly, with a probability of P1/P or P2/P where P1 and P2 are the total number of data points belonging to class 1 and class 2 respectively and P is the total number of data points.
This model can be used as a basis of comparison with more sophisticated models. If the sophisticated model performs worse than a trivial system, then it might indicate that there is a problem with the data modeling. However, a trivial system is not of much help as it doesn't mention the correlation between the input features and the outputs.

3.5.2 Baseline Model: Nearest Means Classifier

In this project, we have implemented the Nearest Means Classifier as our baseline model. The nearest-means classifier works on the following algorithm:

- Calculate the mean of all features for each class.
- The distance between each data point and the class mean is calculated using Euclidean distance.
- The class is assigned depending on which class's mean is closest to the data point.

3.5.3 Logistic Regression

In Logistic Regression, we try to estimate a relationship between the output (dependent variable) and the input (independent variables). It uses a sigmoid function to estimate the probabilities and depending on a particular threshold it assigns the data point to that class.

3.5.4 SVM Classifier: Linear and RBF Kernel

SVM is a supervised learning technique that is used for classification. SVM determines a decision boundary that optimizes the difference between two classes. The margin is the distance between the two classes' closest points. By transferring the input characteristics to a higher-dimensional space, where the data becomes linearly separable, SVM can handle both linearly separable and non-linearly separable data. In SVM, the algorithm chooses the best hyperplane to divide the data into classes. The hyperplane is chosen in such a way that the margin between the classes is maximized. The SVM method seeks the hyperplane with the greatest distance to the nearest data points of the two classes. These nearby data points are referred to as support vectors. When the number of features is more than the number of samples and there is a clear margin of separation between the two classes, SVM is useful.

3.5.5 Multi-Layer Perceptron (MLP)

A Multilayer Perceptron (MLP) is an Artificial Neural Network (ANN) that consists of multiple neurons and is a feedforward network. It uses backpropagation to compute the gradients of weights and biases and is used to update them accordingly. For our project, we implemented multiple learning rates along with SGD and Adam as optimizers.

3.5.6 k-Nearest Neighbors Classifier(kNN)

K-Nearest Neighbors is a non-parametric algorithm used for classification. In kNN, the classifier doesn't make any assumptions about the distribution of the model. Instead, it stores all of the training data and makes predictions based on the new point's distance from the training data points. In kNN, the distance metric used for classification is Euclidean distance.

For this model, we used various values of k to estimate which model gave the best accuracy while performing cross-validation.

3.5.7 Random Forest Classifier (non EE559)

The Random Forest Classifier is an ensemble learning approach that builds numerous decision trees and outputs the class that is the mode of the individual trees' classes. Each decision tree is built using a separate collection of training data and a different group of characteristics.

For this classifier, we used the number of trees as a hyperparameter and decided the best model after performing cross-validation.

3.6 Hyperparameter Search

For the above models, we implemented the hyperparameter search for SVM Classifier using RBF kernels, Multi-Layer Perceptron, kNN and Random Forest Classifier. We tried different hyperparameter values during cross-validation to obtain the best model.

Model	Hyperparameter	Values
SVM Classifier(RBF Kernel)	Gamma	[0.01,0.1,1,10]
Multi-Layer Perceptron	Learning Rate and Optimizer	Learning Rate: [0.0001,0.001,0.01,0.1] Optimizer: [SGD, Adam]
kNN	K values	[1,3,50,100]
Common to RFE and UFS	From 92 to 32(Common Features)	75.804%

Table 8: Hyperparameters for each model

4 Analysis: Comparison of Results, Interpretation

4.1 Metrics

- **Accuracy** Accuracy is defined as the ratio of the sum of True Positives and True Negatives to the sum of all instances.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- **F1 Score** F1 Score is defined as the harmonic mean of precision and recall.

$$\textbf{F1 Score} = 2\frac{(Precision)(Recall)}{Precision + Recall} \tag{2}$$

4.2 Results

The accuracy and F1 score of the best model of each classifier has been tabulated as follows:

Model	Accuracy	F1 Score
Trivial System	50.794%	43.790%
Baseline Model	60.346%	53.646%
Logistic Regression	77.965%	75.149%
SVM Classifier- Linear Kernel	74.95%	71.277%
SVM Classifier- RBF Kernel	98.766%	98.60%
Multi Layer Perceptron	99.203%	99.093%
K Nearest Neighbors	99.236%	99.128%
Random Forest Classifier	**99.53%**	**99.464%**

Table 9: Metrics of the Best Model for each classifier

The best hyperparameter for each model has been tabulated below:

Classifier	Hyperparameter	Best Value
SVM Classifier- RBF Kernel	Gamma	1
Multi Layer Perceptron	Learning Rate and Optimizer	Learning Rate 0.001 with Adam Optimizer
K Nearest Neighbors	Number of Neighbors	1
Random Forest Classifier	Number of Estimators	100

Table 10: Best Hyperparameters for each classifier

The following are the plots obtained:

- **Cross Validation Plots**

  The Cross Validation Plots have been mentioned for only the following classifiers as these were the ones that had provided higher accuracy compared to the rest.

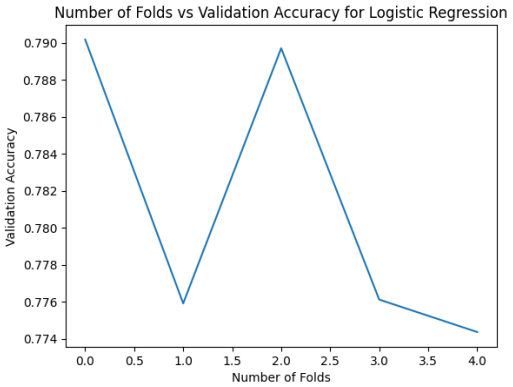

Figure 4: Logistic Regression

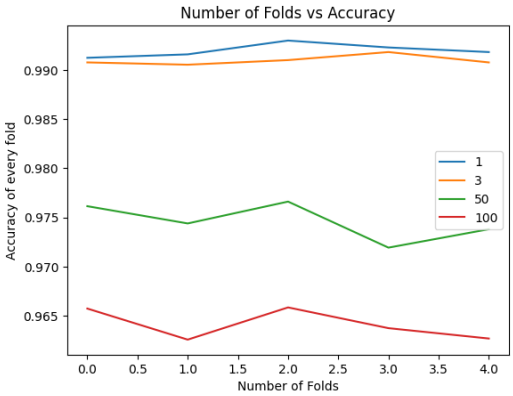

Figure 5: k-Nearest Neighbors

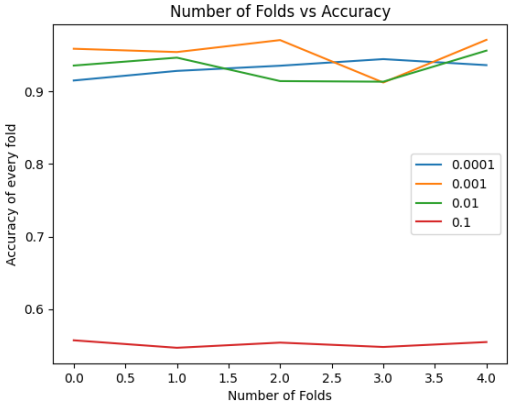

Figure 6: MLP with SGD

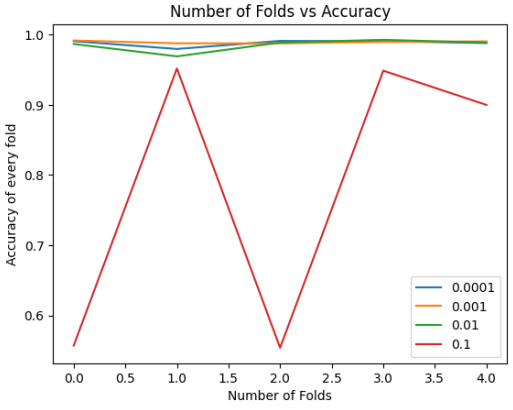

Figure 7: MLP with Adam

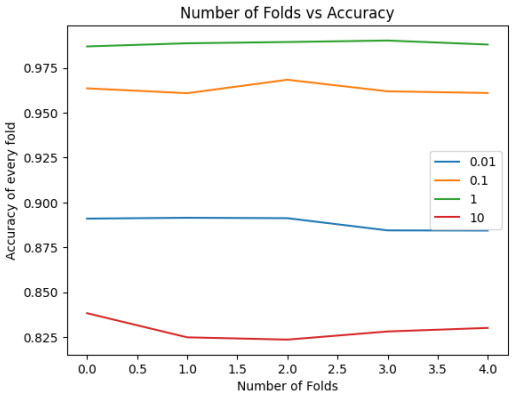

Figure 8: SVM Classifier with RBF Kernel

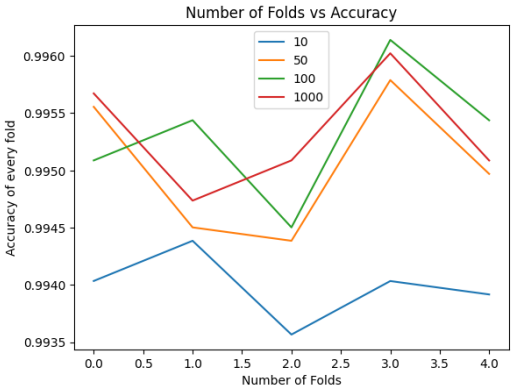

Figure 9: Random Forest Classifier

- **Hyperparameter vs Training and Testing Accuracy Plots**

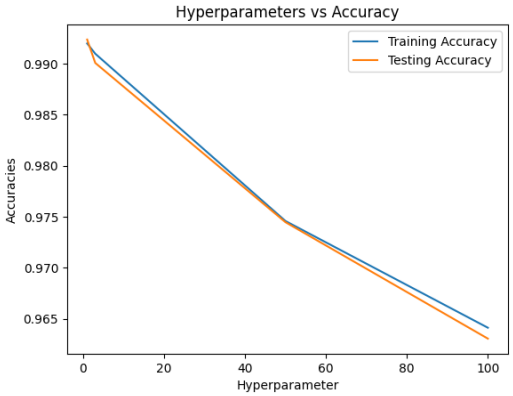

Figure 10: K-Nearest Neighbors

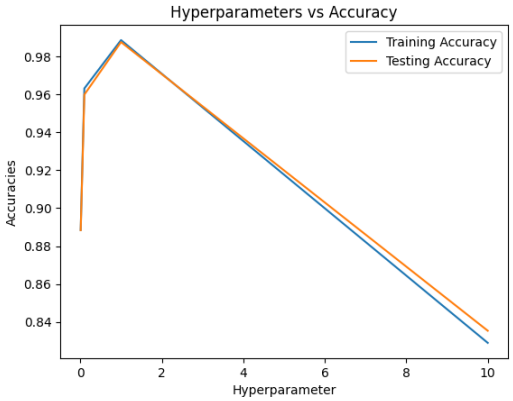

Figure 11: SVM Classifier with RBF kernel

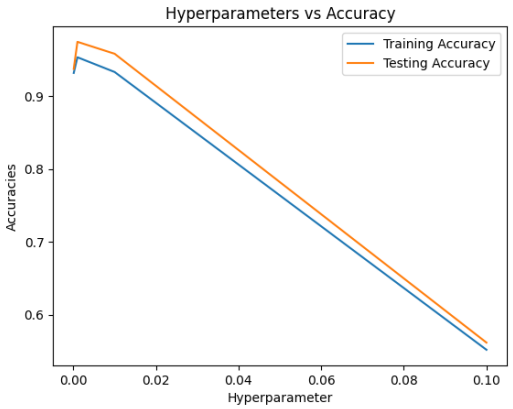

Figure 12: MLP with SGD

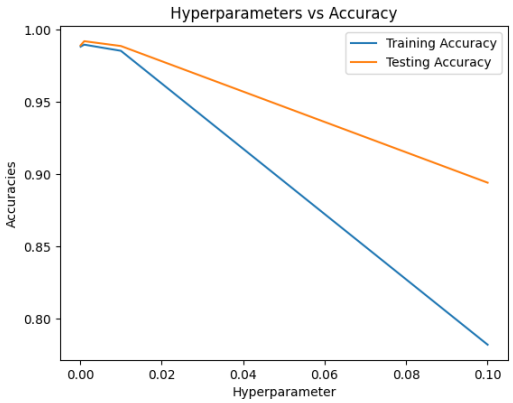

Figure 13: MLP with Adam

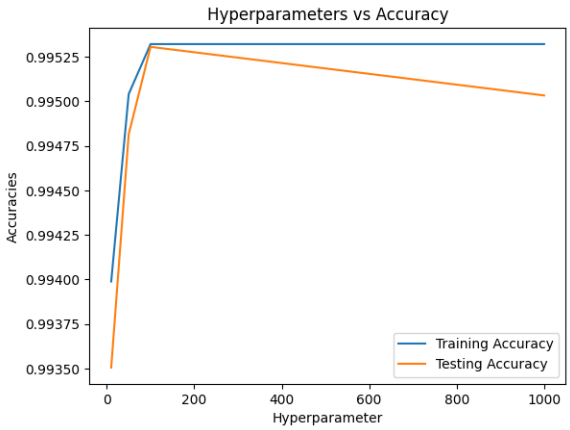

Figure 14: Random Forest Classifier

Observing the accuracy and F1 Score of all the classifiers, Random Forest Classifier with number of estimators as 100 provides the best accuracy and F1 Score. The reason for this is because a random forest classifier uses decision trees to make predictions and having multiple decision trees reduces overfitting. Adding to this, a random forest classifier handles high-dimensional data with non-linear relationships better than other classifiers and captures the complex patterns to make accurate predictions.

Along with this, we were also able to determine the features that are most common to poisonous mushrooms. If a mushroom satisfies more than 1 of the following properties then it can be assumed to be poisonous:

- cap-shape : o
- stem-color : r/b
- gill-attachment : p
- habitat : w
- cap-surface : h
- does-bruise-or-bleed : a
- ring-type : l
- cap-color : l

5 Libraries used and what you coded yourself

Libraries Used	**Coded Ourselves**
Numpy	K-Fold Cross Validation with Split
Pandas	Plots and One Hot Encoding
matplotlib.pyplot	Hyperparameter search for classifiers
sklearn	Trivial System and Nearest Means Classifier

Table 11: Libraries Used and Functions built from scratch

6 Contributions of each team member

Contributions	Contributors
Preprocessing	Vishal
Feature Engineering	Vishal
Cross Validation	Meghana
Classifiers and Hyperparameter Search	Meghana
Report	Vishal and Meghana

Table 12: Individual Contributions

7 Summary and conclusions

Thus, Random Forest Classifier with the number of decision trees as 100 provides the best accuracy of 99.53

References

[1] `https://archive-beta.ics.uci.edu/dataset/73/mushroom`

[2] *Discussion-13 Notes*, at `https://piazza.com/class_profile/get_resource/lcmnuj1ve667jv/lgpwbjp4xng4ly`