Enhancing Interpretability in Mushroom Classification: A Comparative Analysis of Multiple Machine Learning Models

Data Set: Mushroom Dataset

Venkat Giri Anantapatnaikuni
anantapa@usc.edu

Rohit Veeradhi
veeradhi@usc.edu

May 2, 2023

1 Abstract

This study explores various classification models on the mushroom dataset by implementing different feature extraction methods. Three known feature extraction methods and an uncommon method using correlation were used. The performance of these models was evaluated, and the best feature extraction method from the known methods was compared with the correlation method. The results indicate that Trivial System and Naive Bayes performed poorly, while MLP, RF, and SVM had the highest accuracy scores with a minor difference between them. Overall, the Random Forest model achieved the best performance with an accuracy score of 99.907% and an F-1 score of 0.999 for the UFS method. The SVM model achieved the best performance with an accuracy score of 97.019% and an F-1 score of 0.973 for the correlation method. The findings confirm the expected behavior of the models.

2 Introduction

Mushrooms are fungi that produce spores like plants and have a fruit-bearing body. They vary in shape, size, and color. With over 12,000 known species and many more yet to be discovered, they can be classified as edible or poisonous. Some known edible ones are Button and Portobello. For a long time, mushrooms have been essential ingredients in the food industry due to their unique flavors and nutritional properties. Sometimes it is possible to distinguish edible and poisonous mushrooms based on some known visual characteristics, but it can be very misleading. For example, the death cap mushroom, *Amanita Phalloides*, is the most poisonous and be easily mistaken as an edible. This raises the need for some way to classify such mushrooms as they can potentially prevent life-or-death situations. To do this task successfully, extensive research has been done on different AI and ML models. In this project, we analyze the performance of models, namely Trivial System, Baseline System, Logistic Regression(LR), Support Vector Machine(SVM), Naive Bayes(NB), Random Forest(RF), Multi-Layer Perceptron(MLP), and XgBoost on a preprocessed mushroom dataset.

3 Problem Assessment and Goals

As mentioned above, our main goal is to analyze the above-mentioned machine learning models on the given dataset and choose the best model. The given dataset has many categorical values compared to the numerical values, so encoding these values must be done. The reason behind encoding is to extract the important features based on some statistical value, leading to the dimensionality

reduction in the feature set. We aim to perform one-hot encoding and apply different feature elimination methods. We then choose the best feature elimination method to apply different models and analyze the results.

And so the rest of the report is divided as follows: We discuss how the data is distributed and how we split the train dataset into training and validation sets for the cross-validation process. We have presented different feature extraction methods, and from this, we have taken the two best methods and analyzed different models. Finally, we produce our results for each model providing the accuracy and F-1 score.

4 Dataset Description

This task is implemented on a dataset that has characteristics of 173 different mushroom species. This dataset is classified as edible or poisonous. The dataset contains 61,069 data points with 15 unique features, three being metric and the remaining nominal, and one column containing the class labels. The dataset is divided into train and test datasets, with the train set having 42,748 and the test set having 18,321 data points, respectively, and no missing values exist in both dataset.

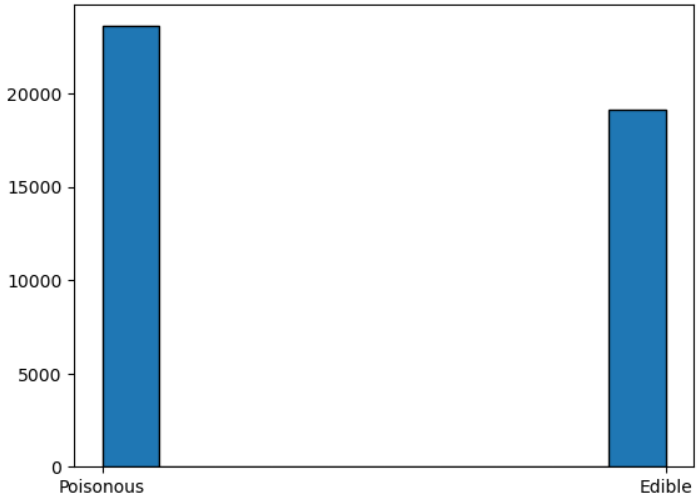


Figure 1: Histogarm of Mushroom Class

From Figure 1, the number of data points classified as poisonous accounts for 23,595 points (55.2%), and edible accounts for 19,153 points (44.8%) of the training dataset. Hence, we can say that the dataset is slightly well-balanced as the probability of occurrence of data points being classified into one of these classes is nearly half.

5 Approach and Implementation

The following sections will dive deeper into the dataset analysis and the preprocessing (one-hot encoding only as the data is already preprocessed). Various common feature reduction methods are described, including a non-common method: we will call this as the Correlation method for this work only. Later, using the best two methods, the mentioned models are analyzed, and the results are put forward.

5.1 Dataset Usage

As mentioned in Section 3, we aim to perform feature engineering and elimination and then apply the mentioned models. The nominal data points are first one-hot encoded, then feature elimination

Features	Unique Nominal Values
cap-shape	7
cap-surface	11
cap-color	12
does-bruise-bleed	2
gill-attachment	7
gill-spacing	3
gill-color	12
stem-color	13
has-ring	2
ring-type	8
habitat	8
season	4
Total	89

Table 1: Count of unique nominal values for each categorical feature

is performed. The categorial features are standardized. These methods are further described in Sections 5.2 and 5.3.

We have defined the number of folds as 10 (KFold = 10) for the cross-validation process. For each fold, we divided the training dataset into training and validation sets having 38,473 and 4,275 data points, respectively. The test dataset was unchanged in terms of data points. For each model, a cross-validation process was performed once. The validation accuracy was calculated for each fold, and the best model corresponding to the highest accuracy obtained was used on the test dataset.

Features	Data Point Coverage for each Unique Label
cap-shape	**c**: 1,288, **x**: 18,808, **s**: 5,079, **o**: 2,417, **f**: 9,289, **b**: 4,043, **p**: 1,824
cap-surface	**i**: 1.599, **y**: 4,460, **t**: 15,569, **d**: 3,125, **g**: 3,280, **h**: 3,482, **k**: 1,602, **s**: 5,341, **e**: 1,789, **w**: 1,494, **l**: 1,007
cap-color	**y**: 6,026, **r**: 1,227, **n**: 16,960, **w**: 5,402, **g**: 3,072, **k**: 907, **o**: 2,581, **l**: 580, **e**: 2,781, **u**: 1,181, **b**: 870, **p**: 1,161
does-bruise-bleed	**f**: 35,279, **t**: 7,469
gill-attachment	**a**: 15,844, **d**: 7,133, **p**: 4,185, **x**: 5,174, **e**: 3,998, **f**: 2,501, **s**: 3,913
gill-spacing	**c**: 34,791, **d**: 5,456, **f**: 2,501
gill-color	**n**: 6,742, **w**: 12,893, **g**: 2,860, **y**: 6,743, **p**: 4,178, **k**: 1,697, **f**: 2,501, **b**: 662, **e**: 756, **o**: 2,041, **r**: 962, **u**: 713
stem-color	**w**: 15,996, **y**: 5,547, **n**: 12,678, **r**: 383 ,**o**: 1,512 ,**g**: 1,841, **l**: 150, **e**: 1,431, **u**: 1,031, **f**: 745, **k**: 599, **p**: 710, **b**: 125
has-ring	**f**: 32,085, **t**: 10,663
ring-type	**f**: 35,581, **r**: 1,013, **e**: 1,703, **z**: 1,418, **l**: 998, **p**: 907, **g**: 888, **m**: 240
habitat	**d**: 30,945, **h**: 1,444, **g**: 5,486, **l**: 2,244, **m**: 2,041, **w**: 259, **p**: 245, **u**: 84
season	**a**: 21,053, **u**: 16,065, **w**: 3,669, **s**: 1,961

Table 2: Count of unique nominal values for each categorical feature

5.2 Preprocessing

The first step is to encode the nominal values. This was done using one-hot encoding techniques where each categorical feature is binary encoded with respect to its corresponding unique labels.

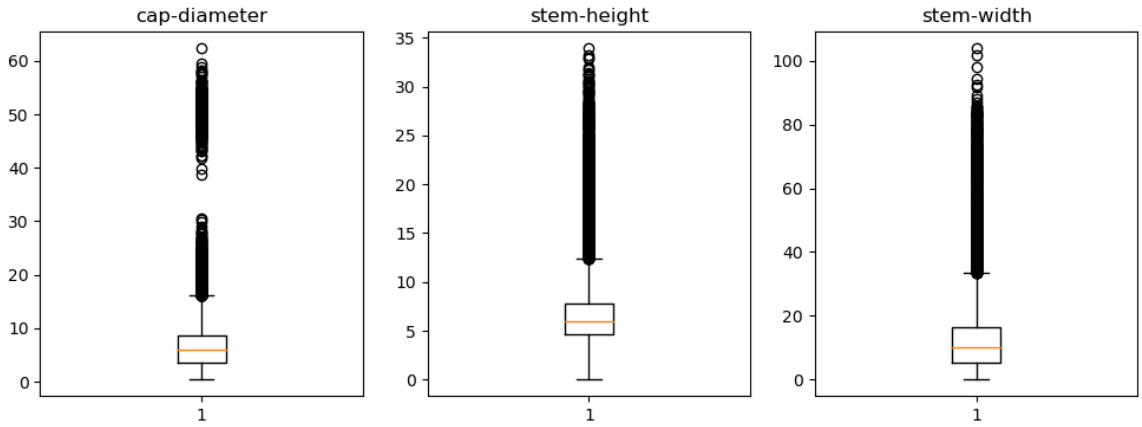
Table 1 shows the number of unique values for each categorical feature. In each feature, the following were the number of data points for each unique label:

From Table 2, after encoding each unique value, a total of 77 features gets added, and the original categorical features are removed. Our training and testing dataset now has 92 features (excluding the class column)[2].

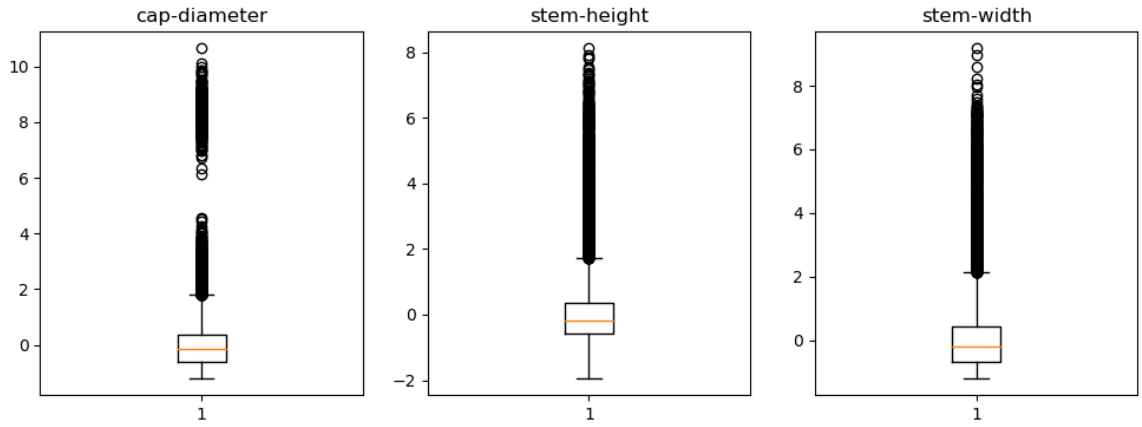
Our next step is to scale the values because there were many outliers in the original metric features. Figure 2 shows this. The scaling was done based on the standardization formula given by the following equation:

$$z_i = \frac{x_i - \mu_i}{\sigma_i^2}$$

where μ_i and σ_i^2 are the ith column's mean and standard deviation respectively. The next step is to apply different feature engineering methods and reduce feature dimensionality.



(a) Original Data



(b) Standardized Data

Figure 2: Box Plot of Cap- Diameter, Stem-Height and Stem-Width before and after Standardization

5.3 Feature Engineering and Dimensionality Adjustment

As mentioned in Section 5.2, the feature engineering method we used is one-hot encoding, as this is the most suitable approach for a dataset dominated by categorical values. A total of 4 different

feature elimination methods have been performed to reduce the feature dimensionality, which has been explained in brief in the following subsections.

5.3.1 Principal Component Analysis (PCA)

Principal Component Analysis is a common method used for feature reduction where the dataset feature dimensionality is large. The process involves generating principal components of the features by calculating the covariance matrix of the data points and performing the eigen decomposition method. The eigen vectors from this method are sorted according to their eigen values in descending order, and the top k vectors will be the new k dimensions.

We have performed PCA on the dataset iteratively by passing all the features with a step size equal to 10. This resulted in a total of 9 iterations. The following steps were performed in each i^{th} iteration, where i = 1, 2, 3....9.

Step 1: The train and test dataset was reduced to i dimensions using PCA.
Step 2: The new fitted data is passed as the input to the Logistic Regression model.
Step 3: Both train and test accuracies were calculated.

After the nine iterations, the accuracies were plotted for each i selected features. Hence, the best decomposition value was 31, with the best test accuracy reaching 60.41%.

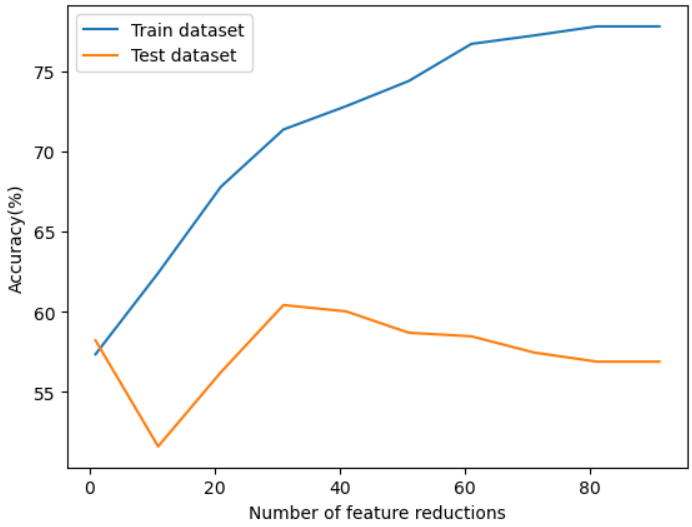


Figure 3: Train vs. Test Accuracy using PCA on Logistic Regression

5.3.2 Univariate Feature Selection (UFS) using Chi-Square

Univariate Feature Selection evaluates each feature using a statistical score and selects the best features most relevant to the target variable (in our case, the class column). The common statistical scores used are Pearson correlation, Chi-Squared test, and ANOVA F-test. We have used the Chi-Squared test to select the features as this method measures the dependence between the categorical features and the target values.

As mentioned in Section 5.3.1, this method was accomplished iteratively by using the same i iteration values. For each iteration, the following steps are performed.

Step 1: A function was defined to perform the Chi-Square test which returns the i+10 selected features.

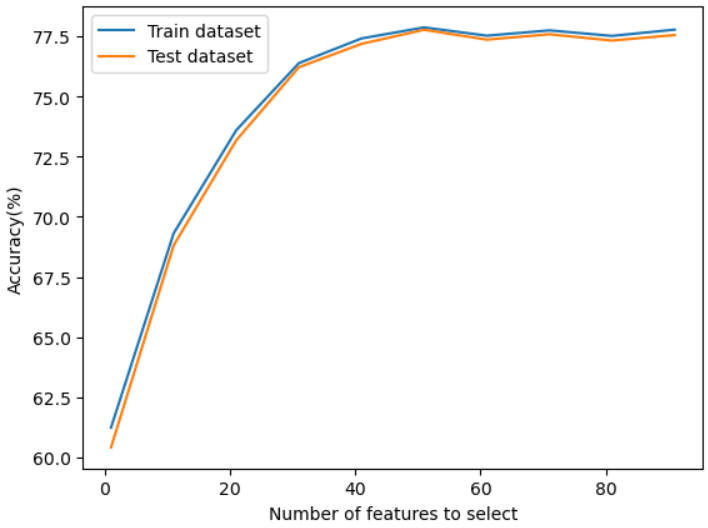


Figure 4: Train vs. Test Accuracy using UFS on Logistic Regression

Step 2: The train and test dataset was generated using these selected features and passed as input to a Logistic Regression model.
Step 3: Both train and test accuracies were calculated.

UFS Feature List	
Features	stem-color_w, gill-attachment_p, stem-width, ring-type_z, stem-height, cap-surface_k, cap-shape_b, gill-color_w, gill-attachment_a, cap-diameter, cap-color_e, stem-color_f, gill-color_n, cap-color_r, cap-color_n, gill-attachment_e, gill-spacing_d, cap-surface_i, cap-shape_o, season_w

Table 3: Best 20 of 51 Features using UFS

After the nine iterations, the accuracies were plotted for each i selected dimensions. For every iteration, we stored the best number of features selected, and the best feature set was selected based on the maximum test accuracy. This resulted in 51 features being selected as the best set with a corresponding test accuracy of 77.76%. A few selected features are shown in Table 3.

5.3.3 Recursive Feature Elimination (RFE)

Recursive Feature Elimination removes features recursively by building a model on each feature. The process involves selecting a model to train the dataset based on the number of features defined. Based on some scores, these features are ranked, and the least features are discarded. This process is repeated until a desired number of features is reached. The model performance is evaluated using the newly selected features.

RFE Feature List	
Features	stem-height, cap-color_u, cap-color_r, gill-color_r, gill-color_p, gill-color_o, cap-surface_s, ring-type_g, ring-type_f, has-ring_f, stem-color_y

Table 4: Best Features using RFE

As mentioned, we again performed this process by iteratively providing ten additional features

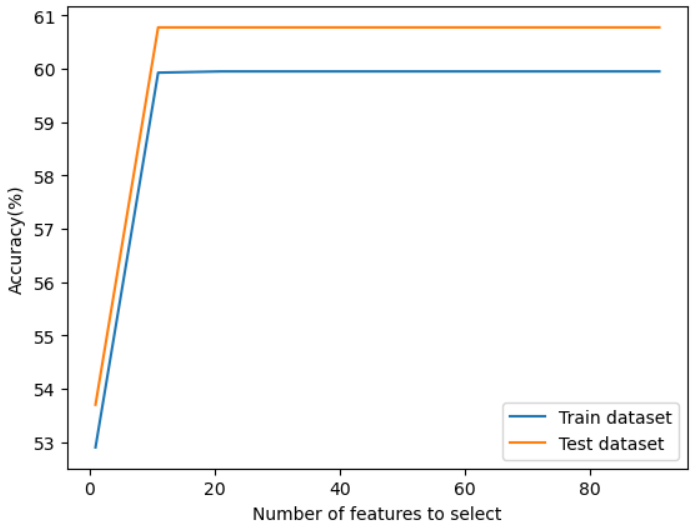


Figure 5: Train vs. Test Accuracy using RFE on Logistic Regression

in each ith iteration. In each iteration, we performed RFE on these features. The following steps are followed.

Step 1: First, we pass the required number of features to be selected in iteration i.
Step 2: For each of these features, we performed Logistic Regression and evaluated the test accuracy. The scores for each of these features are stored.
Step 3: Using these scores, we select the corresponding features and create a new dataset with these feature data points for both the train and test datasets.
Step 4: Both train and test accuracies were calculated.

After the nine iterations, the accuracies were plotted for each i selected dimensions. For every iteration, we stored the best number of features selected, and the best feature set was selected based on the maximum test accuracy. This resulted in 11 features being selected as the best set with a corresponding test accuracy of 60.77%. A few selected features are shown in Table 4.

5.3.4 Feature Reduction using Pearson Correlation (Correlation Method)

Pearson Correlation Coefficient measures the linear dependence between two variables. The values range from -1 to +1, with (+) indicating a strong positive relationship and (-) indicating a strong negative relationship, and 0 indicating no relation. This coefficient measures how closely two variables are related by calculating their degree of association. The formula is given by,

$$r = \frac{\Sigma(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\Sigma(x_i - \overline{x})^2 \Sigma(y_i - \overline{y})^2}}$$

where,

$$\begin{gathered} r = \text{correlation coefficient} \\ x_i = \text{values of the x-variable in a sample} \\ \bar{x} = \text{mean of the values of the x-variable} \\ y_i = \text{values of the y-variable in a sample} \\ \bar{y} = \text{mean of the values of the y-variable} \end{gathered}$$

For our dataset, we found the correlation of all features with respect to class column and selected those features whose correlation values were above 0.1 in magnitude. and the remaining features were discarded. A new train and test dataset was created based on these features.

A total of 22 features were obtained with values more than 0.1 in magnitude, shown in Table 5.

Feature List	
Features	cap-diameter, stem-height, stem-width, cap-shape_b, cap-shape_o, cap-surface_i, cap-surface_k, cap-color_b, cap-color_e, cap-color_n, cap-color_r, gill-attachment_a, gill-attachment_e, gill-attachment_p, gill-spacing_d, gill-color_n, gill-color_w, stem-color_f, stem-color_w, ring-type_z, habitat_g, season_w

Table 5: Best Features using Correlation

From the above four feature reduction methods, we have selected UFE as one of the best feature engineering methods compared to PCA and RFE. This method is compared to the feature reduction method using correlation on each model described in Section 5.4. From these two methods, the best model is chosen.

5.4 Training, Classification, and Model Selection

As mentioned in Section 2, six models have been performed: LR, SVM, NB, RF, MLP, and XgBoost[3]. For each model, one cross-validation run is executed on the train dataset. Since there are ten folds to perform, the accuracy was calculated on the validation set on each fold, and the best model parameters were stored based on the best validation accuracy value. Using the best model parameters, the accuracy on the test dataset was calculated, and this was considered as the final test accuracy for that model.

5.4.1 Trivial System

In this particular model, when a new or "unseen" data point is encountered, it is assigned to one of two classes based on a random probability. This probability is determined by the ratios of N_0/N and N_1/N, where N_0 represents the total number of points belonging to class 0 in the training dataset, N_1 represents the total number of points belonging to class 1, and N is the total number of data points in the training dataset, which is equal to the sum of N_0 and N_1.

For the dataset used, $N_0/N = 0.551$ and $N_1/N = 0.449$.

5.4.2 Nearest Means (Baseline System)

The nearest means model computes two means: one for class 1 using data points classified into class 1 and another for class 0 using data points classified into class 0. To classify a new data point, its Euclidean distance from each mean is calculated, and the data point is assigned to the class of the mean with the shortest distance to it.

5.4.3 Logistic Regression (LR)

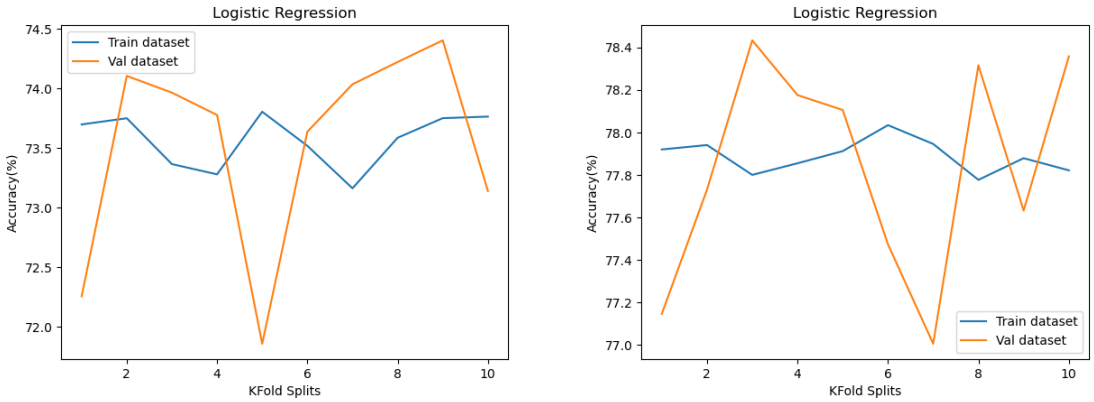
Logistic regression uses logit functions that help derive a relationship between the dependent and independent variables by predicting the probability or probability of occurrence. The logistic functions (sigmoid functions) convert the probabilities into binary values, which could be further used for predictions. We know that the probability of output being one given its inputs can be represented as $P(y = 1|x)$. This, when predicted via linear regression, is expressed as

$$p(X) = \beta_0 + \beta_1 X$$

where, β_0 is the y - intercept and β_1 is slope. Applying the logit function to the above equation and taking the inverse of it gives the logit function given by

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

We have performed logistic regression using the features from UFE and Correlation feature reduction. Figure 6 was obtained and showed the accuracy differences between the two reduction methods.



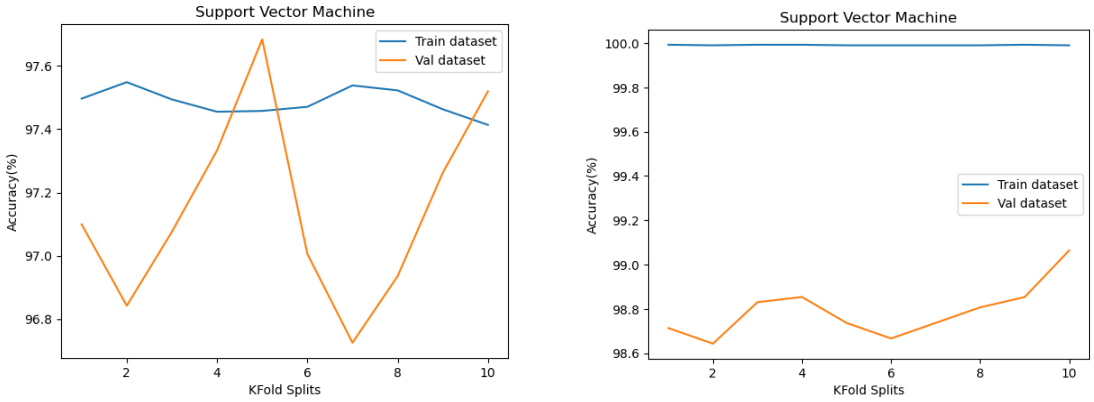
(a) Feature Reduction using Correlation (b) Feature Reduction using UFE

Figure 6: Train vs. Validation Accuracy using Logistic Regression

5.4.4 Support Vector Machine (SVM)

Support Vector Machine is a discriminative machine learning algorithm used for classification. It separates data points into two classes at a time using a decision boundary, typically a hyperplane. SVM aims to find the best possible decision boundary that separates the classes, known as the Optimal Separating Hyperplane. It's worth noting that SVM can handle more than two classes but still separates data into two classes at a time[7].

$$\phi(X) = exp(\frac{-x^2}{2\sigma^2}), \sigma > 0$$



(a) Feature Reduction using Correlation (b) Feature Reduction using UFE

Figure 7: Train vs. Validation Accuracy using SVM

From Section 5.5, Gaussian Radial Basis (RBF) kernel is used with $Gamma = 10$ and $C = 3$. where C is the regularization parameter which controls the trade-off between achieving a low training

error and a low testing error, and Gamma is the parameter that controls the shape of the decision boundary[8][9]. We have performed SVM using the features from UFE and Correlation feature reduction. Figure 7 was obtained and showed the accuracy differences between the two reduction methods.

5.4.5 Naive Bayes (NB)

Naive Bayes is a commonly used machine learning technique for solving classification problems. It relies on Bayes' theorem and assumes that the features are conditionally independent. The algorithm estimates the probability of each class based on the input features and selects the class with the highest probability as the predicted class[7]. Using the chain rule, the likelihood $P(x|y)$ can be written as,

$$P(x_i|y, x_1, ..., x_{i-1}, x_i, ..., x_n) = P(x_i|y)$$

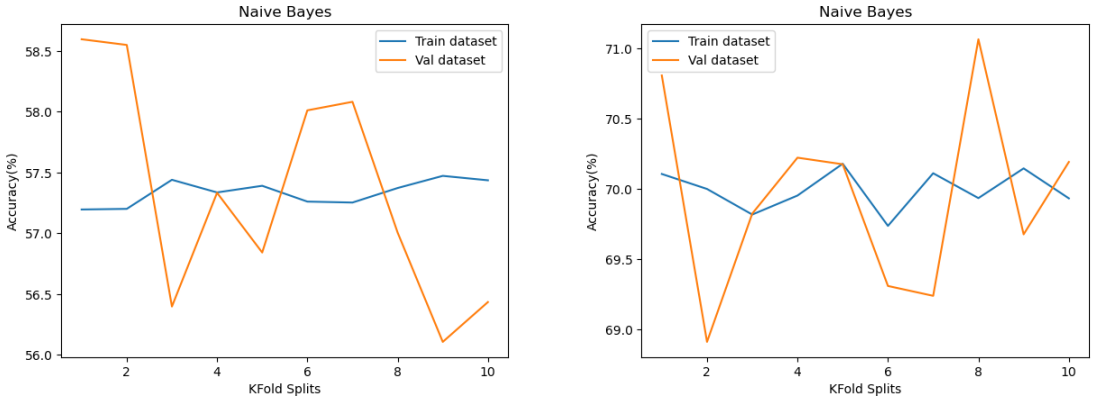
$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ...x_n)}$$

where, $P(y|x_1, ..., x_n)$ is defined as the posterior probability, $P(y)$ is the likelihood, and $P(x_|y)$ is the prior. The denominator can be considered a constant for a given input, so we can further simplify the equation and use it for classification.

$$P(y|x_1, ..., x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

The Naive Bayes classifier applies a decision rule to the model it creates. One commonly used decision rule is the maximum a posteriori (MAP) decision rule[9], which involves selecting the hypothesis with the highest probability, which is given by,

$$\hat{y} = \arg\max_{y} P(y)\prod_{i=1}^{n} P(x_i|y)$$



(a) Feature Reduction using Correlation (b) Feature Reduction using UFE

Figure 8: Train vs. Validation Accuracy using NB

We have used GaussianNB(), which estimates the Gaussian (normal) distribution for each input feature and class. It applies Bayes' theorem to compute the probability of each class given the input

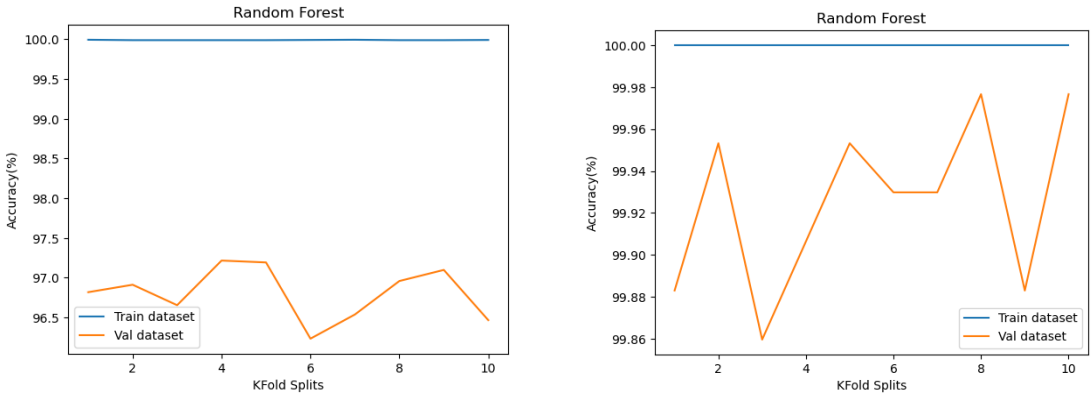
features and subsequently identifies the class with the highest probability as the predicted class[8]. Figure 8 was obtained and showed the accuracy differences between the two reduction methods.

5.4.6 Random Forest (RF)

Random Forest is a supervised machine learning algorithm that can be used for classification and regression tasks. In this project, we're utilizing Random Forest for classification purposes. The algorithm creates decision trees using different data and utilizes a majority vote for classification. We can specify the number of decision trees we want to use for classification. Random Forest uses an ensemble technique called bagging, combining multiple classifiers to make predictions[8]. Bagging selects random data points from the dataset with replacement, trains each tree independently, and generates results. The final classification is made by taking a majority vote based on the combined results of all the classifiers. Below are the following steps:-

Step 1: Randomly select k data points from the training dataset.
Step 2: Create an individual decision tree using each k-selected data point.
Step 3: Each decision tree generates a corresponding output.
Step 4: The final classification is made by taking a majority vote based on the outputs of all the decision trees.

In our project, we utilized the ensemble RandomForestClassifier module of sklearn to specify the number of estimators to 100 directly. Figure 9 was obtained and showed the accuracy differences between the two reduction methods.



(a) Feature Reduction using Correlation (b) Feature Reduction using UFE

Figure 9: Train vs. Validation Accuracy using RF

5.4.7 Multi-Layer Perceptron (MLP)

A multi-layer perceptron is an artificial neural network (ANN) composed of multiple layers of interconnected nodes. Each node receives an input from the previous layer, applies an activation function to that input, and produces an output that is then passed on to the next layer. A simple three-layer network consists of a first layer, i.e., the input layer, a last, i.e., the output layer, and a middle layer, referred to as the hidden layer. Every node generally has an activation function. Sometimes referred to as non-linearities, this enables input-output relationships to be described in a non-linear manner and enhances the model's ability to be adaptable in describing complex relationships. Perceptrons use real-valued inputs to generate a singular output through a linear combination that involves input weights. This output may also be subjected to a non-linear activation function. To express this mathematically:

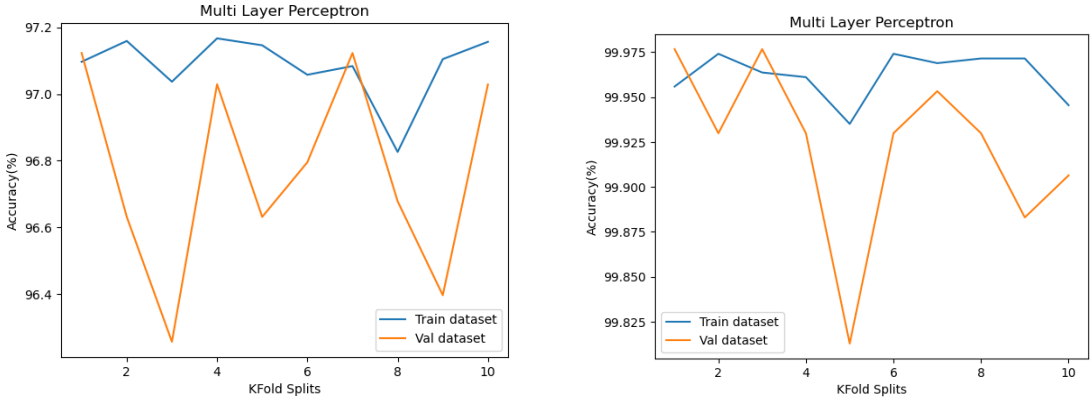
$$y = \varphi(\sum_{n=1}^{\infty} w_i x_i + b) = \varphi(w^T x + b)$$

where w denotes the vector of weights, x is the vector of inputs, b is the bias and φ is the non-linear activation function. Training the model happens in three stages: First is a forward pass where the input to the model is multiplied with weights, and bias is added at every layer to find the calculated output of the model. The second is the loss calculation based on the predicted and actual true output. Finally, a backward pass, commonly known as back propagation, is performed to update the weights and bias using gradient.

For this dataset, we have utilized MLPClassifier module of sklearn to specify different parameters of the ANN network. The ANN network consists of 3 layers with the following parameters from Section 5.5:

- **Hidden Layers**: We have initialized 100 neurons to one hidden layer. This was taken arbitrarily.
- **Activation Function**: Rectified Linear Unit (ReLU) function is used as it is known to work well for various problems.
- **Optimizer**: We have used Adam Solver for its fast convergence and low memory requirements.
- **Regularizer**: We have use L2 regularization to prevent overfitting. A small value of 0.0001 is chosen.
- **Learning Rate**: A value of 0.001 is chosen to ensure the stable convergence of the model.

Figure 10 was obtained and showed the accuracy differences between the two reduction methods.



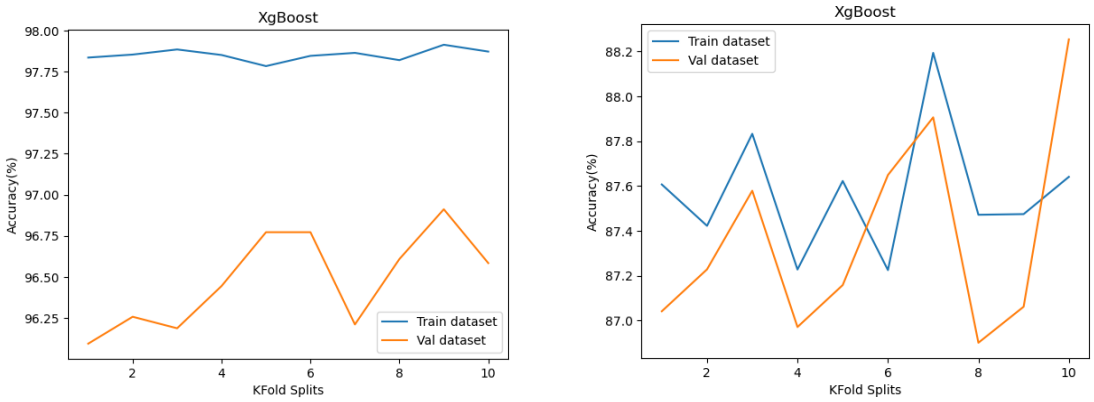
(a) Feature Reduction using Correlation (b) Feature Reduction using UFE

Figure 10: Train vs. Validation Accuracy using MLP

5.4.8 XgBoost

XGBoost, also known as eXtreme Gradient Boosting, is a potent and widely used machine learning algorithm that falls under the category of boosting algorithms. It is specifically engineered to handle structured data in a tabular format and is applicable for solving regression, classification, and ranking problems.

The algorithm follows an ensemble learning approach where it iteratively builds multiple decision trees. Each iteration assigns more weight to the incorrectly classified samples and generates a



(a) Feature Reduction using Correlation (b) Feature Reduction using UFE

Figure 11: Train vs. Validation Accuracy using XgBoost

new decision tree to correct the previous iteration's errors. The algorithm aggregates all the trees' predictions to obtain the final prediction. Figure 11 was obtained and showed the accuracy differences between the two reduction methods based on the parameters obtained from Section 5.5

5.5 Parameter setup for SVM, MLP and XgBoost

This section is divided into two parts. The first corresponds to the UFS method, while the second is the correlation method. The training dataset was split into train and validation sets, with the validation set having 33% of data points from the train set. We heuristically tune the hyperparameters of these classifiers until we achieve satisfactory performance on the validation set. Validation accuracy, as well as the F-1 score, was calculated. The F1 score is calculated as

$$F-1score = \frac{2 * precision * recall}{precision + recall}$$

where

$$precision = \frac{truepositive}{truepositive + falsepositive}, recall = \frac{truepositive}{truepositive + falsenegative}$$

.

1. **UFS Method**: Here we present the accuracy and F-1 scores using UFS method.

SVM	**Validation accuracy (%)**	**F1 score**
kernel = 'linear', C = 0.5	75.6	0.775
kernel ='rbf', C = 0.5, gamma = 3	98.45	0.985
kernel = 'rbf', C = 3 gamma = 10	**99.734**	**0.997**

XgBoost	Validation accuracy (%)	F1 score
max_depth = 3, eta = 0.1, gamma = 1	87.523	0.886
max_depth = 8, eta = 0.5, gamma = 3	**99.723**	**0.997**
max_depth = 15, eta = 0.8, gamma = 7	99.624	0.996

MLP	Validation accuracy (%)	F1 score
activation = 'relu', solver = 'adam', alpha = 0.0001, learning_rate_init = 0.001	**99.936**	**0.999**
activation = 'logistic', solver = 'sgd', alpha = 0.001, learning_rate_init = 0.01	99.496	0.995
activation = 'tanh', solver = 'adam', alpha = 0.01, learning_rate_init = 0.5	75.522	0.717

2. **Correlation Method**: Here we present the accuracy and F-1 scores using correlation method.

SVM	Validation accuracy (%)	F1 score
kernel = 'linear', C = 0.5	71.163	0.750
kernel ='rbf', C = 0.5, gamma = 3	96.2	0.965
kernel = 'rbf', C = 3 gamma = 10	**96.93**	**0.972**

XgBoost	Validation accuracy (%)	F1 score
max_depth = 3, eta = 0.1, gamma = 1	84.752	0.869
max_depth = 8, eta = 0.5, gamma = 3	**96.094**	**0.964**
max_depth = 15, eta = 0.8, gamma = 7	96.023	0.964

MLP	Validation accuracy (%)	F1 score
activation = 'relu', solver = 'adam', alpha = 0.0001, learning_rate_init = 0.001	**96.590**	**0.969**
activation = 'logistic', solver = 'sgd', alpha = 0.001, learning_rate_init = 0.01	95.073	0.955
activation = 'tanh', solver = 'adam', alpha = 0.01, learning_rate_init = 0.5	88.112	0.889

From the above tables, we have taken the best parameter values and trained the models in Sections 5.4.4, 5.4.7, and 5.4.8, respectively. We only tuned the parameters for these classifiers as these significantly impacted model performance, whereas the hyperparameters for the remaining models did not.

6 Analysis: Comparison of Results, Interpretation

In this section, we present the results obtained on the test set for each model, along with accuracies and F1 scores.

Classifiers	UFS Method		Correlation Method	
	Accuracy (%)	F-1 score	Accuracy (%)	F-1 score
Trivial System	50.652	0.447	50.412	0.438
Baseline System (Nearest means)	62.856	0.673	62.856	0.673
Logistic Regression	77.757	0.798	73.407	0.763
SVM	98.760	0.988	**97.019**	**0.973**
Naive Bayes	70.334	0.762	56.749	0.395
Random Forest	**99.907**	**0.999**	96.905	0.972
XgBoost	87.265	0.884	96.304	0.967
MLP	99.863	0.998	96.785	0.971

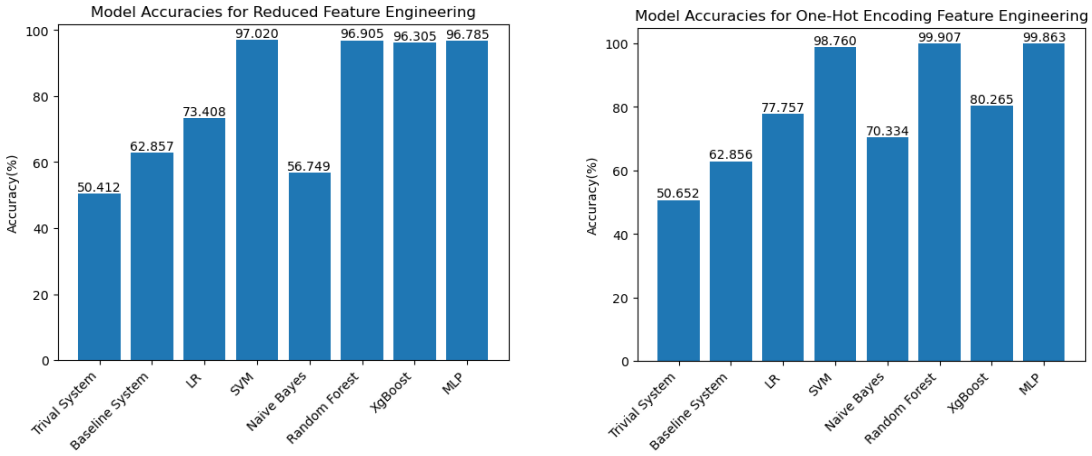
Table 6: Test accuracy and F1 score of all the classifiers used

Table 6 shows that the best model for UFS method is MLP, and for the Correlation method is Random Forest. The worst-performing models were Trivial System and Naive Bayes. As expected, Naive Bayes and Trivial System have the least accuracies for both methods because they use probabilistic algorithms. Since the trivial system assigned the classes for the random choices based on the class probability, Naiver Bayes assumes the features are independent, given the class, which may or may not be true for our dataset.

From Table 2, we observe that many values have a high degree of overlap in each feature which can also cause poor performance. The best models were SVM, RF, XgBoost, and MLP. This was expected as the dataset has a relatively small number of features after feature extraction compared to the number of observations. It is also due to the fact that these models use more complex algorithms compared to the remaining models mentioned and are generally less prone to over-fitting.

In conclusion, the best-performing model for UFS method was Random Forest with an accuracy

of 99.907% with an F-1 score of 0.999, whereas, for the correlation method, it was SVM with an accuracy of 97.019% and an F-1 score of 0.973. The models performed as expected. Figure 12 shows the accuracy bar plots of all models.



(a) Correlation Method (b) UFS Method

Figure 12: Test Accuracies of all models

7 Libraries used and what you coded yourself

The following is a summary of the Python libraries, along with the modules that were coded by the team.

Libraries used	**Modules Coded**
math, random	Feature Engineering and Extraction (using libraries)
numpy, scipy	Cross-Validation setup (using library)
pandas, matplotlib.pyplot	Trivial System (from scratch)
sklearn, xgboost	Nearest-means (from scratch)

8 Contributions of each team member

Work	**Contributors**
PCA, UFS, RFE	Venkat and Rohit
Correlation Method	Rohit
Trivial System and Baseline System	Venkat
LR, MLP, XgBoost, RF	Rohit
Naive Bayes and Support Vector Machine	Venkat
Data Analysis	Venkat and Rohit
Report Writing	Venkat and Rohit

9 Summary and conclusions

In this work, we studied different classification models on the mushroom dataset. Feature extraction was performed based on different methods for the main models, and the best two methods were compared. The models chosen behaved as expected, with Trivial System and Naive Bayes performing the worst and MLP, RF, and SVM performing the best with a negligible difference for both methods. For future work, a potential solution to increase the performance of the low-performing models is to do data augmentation. More ensemble methods can be used, such as bagging, boosting, and stacking, rather than relying on a single machine learning algorithm. Another approach would be to use transfer learning on pre-trained models to improve the overall performance.

References

[1] S. Ismail, A. R. Zainal and A. Mustapha, "Behavioural features for mushroom classification," 2018 IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE), Penang, Malaysia, 2018, pp. 412-415, doi: 10.1109/ISCAIE.2018.8405508

[2] Blog post: Mushroom Classification using Machine Learning with Deployment using FastAPI

[3] Blog post: Mushroom Classification Using Different Classifiers

[4] Wagner, D., Heider, D. Hattab, G., "Mushroom data creation, curation, and simulation to support classification tasks", Sci Rep 11, 8134 (2021). https://doi.org/10.1038/s41598-021-87602-3

[5] Blog post: Leveraging Feature Importance to Predict Mushrooms' Edibility in Python

[6] M. I. Habibie and N. Nurda, "Performance Analysis and Classification using Naive bayes and Logistic Regression on Big Data," 2022 1st International Conference on Smart Technology, Applied Informatics, and Engineering (APICS), Surakarta, Indonesia, 2022, pp. 48-52, doi: 10.1109/APICS56469.2022.9918793.

[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), Coimbatore, India, 2022, pp. 1-7, doi: 10.1109/ICCCI54379.2022.9741040.

[8] M. S. Ahmed et al., "Comparative Analysis of Interpretable Mushroom Classification using Several Machine Learning Models," 2022 25th International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh, 2022, pp. 31-36, doi: 10.1109/ICCIT57492.2022.10055555.

[9] A. Wibowo, Y. Rahayu, A. Riyanto and T. Hidayatulloh, "Classification algorithm for edible mushroom identification," 2018 International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2018, pp. 250-253, doi: 10.1109/ICOIACT.2018.8350746.

[10] Blog post: Using Extreme Gradient Boosted Trees in Machine Learning classification problems