1. Data Cleaning:

In the initial step, the necessary libraries, such as NumPy, pandas, and sklearn, are imported. The dataset is loaded using the **`read_csv()`** function from pandas, which reads the data from the specified file **("/content/mushrooms.csv")**. The **info()** method is then used to obtain an overview of the dataset. From the output, we can see that the dataset contains 8,124 entries with 23 columns, all of which are of object type.

2. Data Preprocessing:

To preprocess the data and make it suitable for analysis, the categorical variables in the dataset need to be converted into a numeric form. The label encoder from sklearn's preprocessing module is used to achieve this. A loop is implemented to iterate over each column in the dataset, and the label encoder is applied to convert the categorical values into numeric labels. This process ensures that the data is in a suitable format for further analysis. The encoded data is then stored back into the respective columns of the dataframe. The **"veil-type"** column is dropped using the **drop()** function, as it contains only a single unique value and does not contribute to the classification task.

3. Splitting Data:

The dataset is split into features and labels. The target variable, **"class"** is assigned to the variable **y**, while the remaining columns are assigned to the variable **X**. The **train_test_split()** function from sklearn's model_selection module is used to split the data into training and testing sets. In this case, the test size is set to 0.2, indicating that 20% of the data will be used for testing, while the remaining 80% will be used for training. Additionally, the training set is further split into training and validation sets using the same function. The test size for this second split is set to 0.25, resulting in a 60:20:20 ratio for training, validation, and testing, respectively.

4. Initialize Decision Tree Model:

An instance of the DecisionTreeClassifier model from sklearn's tree module is created and assigned to the variable **model**. This model will be used to build a decision tree classifier. The decision tree algorithm is a popular choice for classification tasks as it creates a tree-like model of decisions and their possible consequences.

5. Feature Selection:

To identify the most relevant features in the dataset, Recursive Feature Elimination (RFE) is applied. RFE is a feature selection technique that works by recursively eliminating less significant features from the dataset. It starts by fitting the model (DecisionTreeClassifier in this case) on the training data and then eliminates the least important features based on their rankings. This process continues iteratively until the desired number of features is selected. In this code, the **RFE()** function from sklearn's feature_selection module is used with the **model** (DecisionTreeClassifier) as the estimator and **n_features_to_select** set to 5, indicating that the aim is to select the top 5 features. After applying RFE, the indices of the selected features are obtained using the **get_support(indices=True)** method, and their corresponding names are extracted from the original dataset's columns using the **columns** attribute. The selected features are: **'gill-size', 'gill-color', 'stalk-root', 'spore-print-color', and 'population'.**

6. Data Preprocessing (Removing Rest of Features):

To focus on the selected features, the training, validation, and testing datasets are filtered to include only those features. This is done by assigning the selected feature names to the respective datasets: **X_train_selected**, **X_val_selected**, and **X_test_selected**. These datasets will be used for training, validation, and testing the model, respectively.

7. Training Model:

The decision tree model (**model**) is trained using the **fit()** method. The training data consists of the selected features (**X_train_selected**) and their corresponding labels (**y_train**). By fitting the model on the training data, it learns the patterns and relationships between the selected features and their associated labels.

8. Testing:

The trained decision tree model is then used to make predictions on the validation dataset (**X_val_selected**). The predictions are compared to the actual labels (**y_val**) to evaluate the model's performance. The **classification_report()** function from sklearn's metrics module is used to generate a comprehensive report on the classification results. This report includes metrics such as precision, recall, F1-score, and support for each class (0 and 1).

9. Validation:

The validation results show that the decision tree model performs well on the validation dataset. The accuracy of the model is calculated as 0.99, indicating that

it correctly predicts the class labels for 99% of the validation instances. The precision, recall, and F1-score are high for both classes (0 and 1), which suggests that the model can effectively distinguish between the two classes. The macro average and weighted average of these metrics are also close to 0.99, indicating a balanced performance across the classes. The support values represent the number of instances for each class in the validation set.

10. Deployment:

The decision tree model has been deployed on a web application, allowing users to interact with the model and make predictions using the selected features. The web application link is provided at the bottom. By clicking on the link, users will be directed to the web app interface, where they can input the values for the selected features (**gill-size, gill-color, stalk-root, spore-print-color, and population**). Once the user submits the input, the deployed model will use the provided values to make predictions and display the predicted class label. The web app provides a user-friendly interface, allowing users to easily interact with the model and obtain predictions for mushroom classification based on the selected features. This deployment enables users to utilize the model's capabilities without requiring any programming or technical knowledge.

Link: https://mushroom-model.azurewebsites.net/