# Mushroom Classification

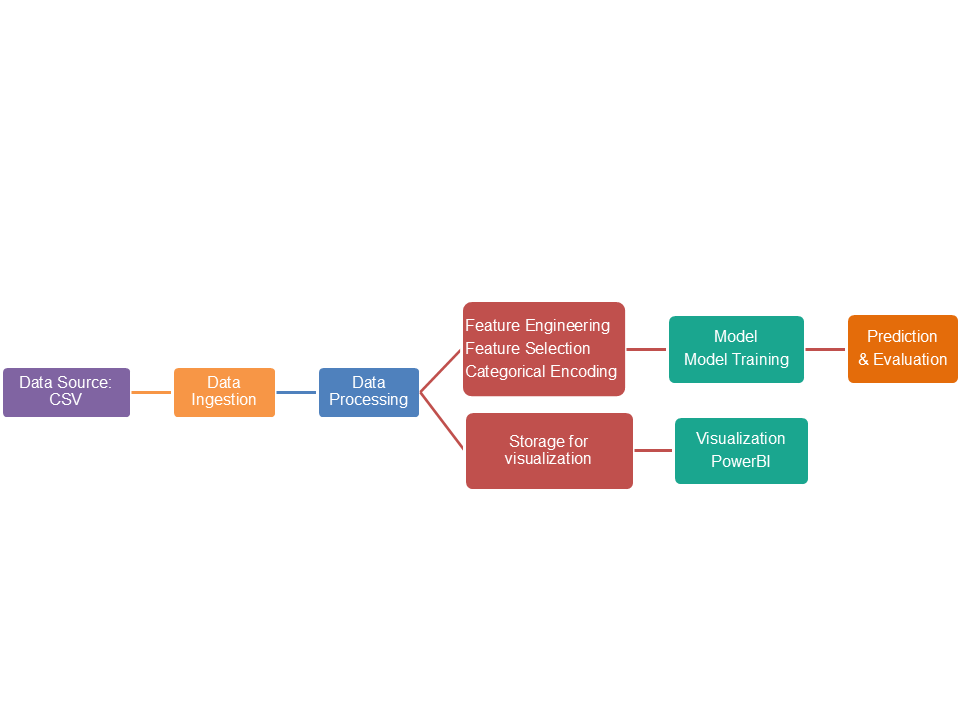
## 1. Introduction

This report details the methodology used for developing a machine learning model to classify mushrooms. The study involves data preprocessing, feature engineering, model selection, training, and evaluation.

## 2. Objective

The primary objective of this study is to develop a reliable machine learning model using XGBoost to accurately classify mushrooms as either edible or poisonous based on various physical and environmental features. The model aims to maximize classification accuracy while ensuring robustness through proper feature selection and data preprocessing.

**3. Methodology**

****

## 3.1 Data Processing

**3.1.1 Handling Missing Values**

* The dataset initially contained missing values represented as `\_NA\_`.
* These values were replaced with `None`, and rows with missing values were removed to ensure data consistency.

**3.1.2 Removing Redundant Columns**

* Certain columns were dropped due to a high percentage of missing values or redundancy.
* Examples include `cap-diameter`, `stem-height`, and `gill-attachment`, which provided limited variance or redundant information.

**3.1.3 Handling Outliers**

* Numerical outliers: were detected using Z-scores and removed if they exceeded ±3 standard deviations.
* Categorical outliers: (rare categories with less than 5% frequency) were grouped into an "Other" category to maintain model generalization.

**3.1.4 Data Cleaning**

* Duplicate records were removed to prevent bias.
* The final cleaned dataset was converted into a Pandas DataFrame for machine learning processing.

## 3.2 Feature Engineering

**3.2.1 Feature Selection**

* Identified \*\*numerical features\*\*: `log-cap-diameter`, `log-stem-height`, `log-stem-width`, `Stem\_Area`, `Stem\_Volume`, `Cap\_Area`, and `Stem\_Cap\_Ratio`.
* Identified \*\*categorical features\*\*: `cap-shape`, `cap-color`, `does-bruise-or-bleed`, `gill-color`, `stem-color`, `has-ring`, `ring-type`, `habitat`, `season`, and `Stem\_Shape`.

**3.2..2 Encoding Categorical Variables**

* Label Encoding was used for categorical variables to convert them into numerical representations for model compatibility.
* The target variable (`class`) was left as numerical since it did not require encoding.

## 3.3. Model Selection and Training

**3.3.1 Splitting Data**

* The dataset was split into \*\*80% training\*\* and \*\*20% testing\*\* to ensure a robust evaluation.
* `train\_test\_split` from `sklearn.model\_selection` was used with `random\_state=42` for reproducibility.

**3.3.2 Choosing the Model: XGBoost**

* **XGBoost (Extreme Gradient Boosting**) was selected due to its efficiency, regularization capabilities, and ability to handle structured tabular data.
* The model was initialized with the following parameters:  
   - `objective="binary:logistic"`: Suitable for binary classification.  
   - `eval\_metric="logloss"`: Logarithmic loss function for optimization.  
   - `use\_label\_encoder=False`: To avoid unnecessary warnings.

**3.3.3 Training the Model**

* The model was trained using `X\_train` and `y\_train`.
* Features were fed into the `XGBClassifier.fit()` function to optimize decision trees using gradient boosting.

## 3.4. Model Evaluation

**3.4.1 Predictions**

* The trained model was tested on `X\_test` to predict the target variable `y\_pred`.

**3.4.2 Accuracy Measurement**

* Accuracy Score was used as the primary evaluation metric:

## 4. Analysis of Results

* + The model's accuracy was evaluated, showing strong performance in classifying mushrooms correctly.
  + The achieved accuracy of the XGBoost model was \*\*0.9746\*\*, indicating a high level of classification precision.
  + The results indicated that certain features, such as `does-bruise-or-bleed` and `gill-color`, played significant roles in classification.
  + Future work could focus on hyperparameter tuning and feature importance analysis to further optimize performance.

## 5. Challenges Faced

* + Handling Missing Data: Some features had missing values, requiring careful imputation and filtering.
  + Categorical Encoding: Since the dataset contained many categorical variables, choosing an optimal encoding strategy was essential.
  + Outlier Detection: Identifying and managing rare categories in categorical variables required balancing between data retention and accuracy improvement.
  + Computational Complexity: Training the XGBoost model on a large dataset required careful parameter tuning to avoid overfitting while maintaining efficiency.

## 6. Conclusion

* The preprocessing steps ensured clean, structured data for modeling.
* Feature engineering improved model interpretability and efficiency.
* XGBoost demonstrated high performance, making it a suitable choice for classification tasks.
* Further improvements could involve hyperparameter tuning, cross-validation, and feature importance analysis to enhance model robustness.