Binary Prediction of Poisonous Mushrooms

Abheesht Roy
ASU ID – 1233225866
CSE 572

Today's agenda

- Background
- Problem Statement
- Challenges
- Data
- Pipeline
- Results
- Next Steps

Background

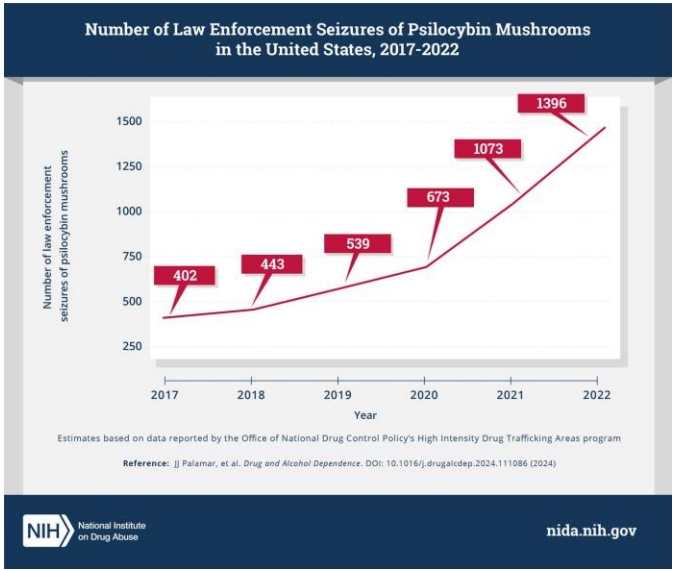
- Mushroom foraging is common worldwide
- Visual identification is difficult & risky
- Mushroom classification is a classic ML dataset
- Kaggle Playground: synthetic but realistic data

Problem Statement

- Build a system to predict whether a mushroom is **edible or poisonous**
- Use physical/categorical features like **cap shape, color, odor, and gill type**
- Formulated as a **binary classification task** with two classes: edible (e) and poisonous (p)
- Competition performance measured using **Matthews Correlation Coefficient (MCC)**

Why It Matters

- **Health & Safety**: Eating poisonous mushrooms can be fatal
- **Visual identification is unreliable** even for experienced foragers
- **Educational value**: Classic dataset for learning classification
- **Data mining practice**: Real-world, high-stakes prediction task

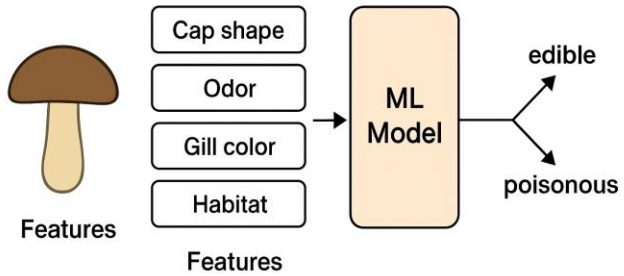


Challenges

- **Categorical features only** → require encoding before modeling (e.g., one-hot, ordinal encoding)
- **Feature explosion & high dimensionality** → many new variables after encoding
- **Possible class imbalance** → edible vs. poisonous distribution may skew results
- **Overfitting risk** → synthetic dataset + many features can cause poor generalization
- **Metric choice (MCC)** → harder to optimize than simple accuracy, must handle all error types
- **Feature redundancy** → some attributes may carry duplicate or low-value information

Problem Formulation

- Task type: **Supervised Binary Classification**
- Input: Mushroom attributes (cap shape, odor, gill color, habitat, etc.)
- Output: Target label → **edible (e) or poisonous (p)**
- Prediction objective: Build a model that **maps features → class label**
- Evaluation: **Matthews Correlation Coefficient (MCC)**
- Formally:
    - Learn function $f(X) \rightarrow y$, where X = feature vector, $y \in \{e, p\}$



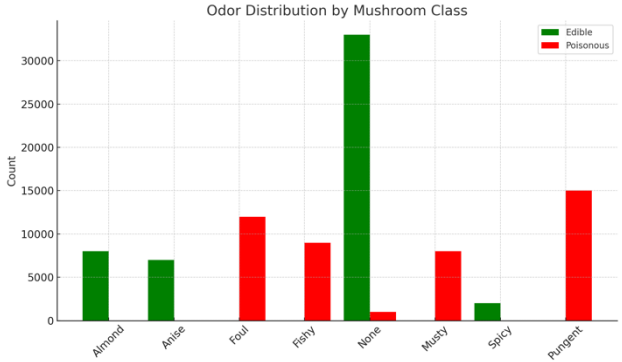
Planned Data Mining Pipeline

- **Data Collection** → Kaggle Mushroom dataset (synthetic, labeled)
- **Data Preprocessing**
  - Handle categorical variables (one-hot/label encoding)
  - Remove/merge redundant features
  - Check for missing values
- **Exploratory Data Analysis (EDA)**
  - Visualize feature distributions & class balance
  - Identify strong predictors (e.g., odor, spore print color)
- **Modeling**
  - Baseline: Logistic Regression, Decision Trees
  - Advanced: Random Forest, XGBoost, Neural Network (optional)
- **Evaluation**
  - Train/test split, cross-validation
  - Metrics: Accuracy, Precision/Recall, **MCC**
- **Comparison & Selection** → choose best-performing model

Initial Data Exploration

- **Initial Observations**
  - Dataset size: ~95,000 rows, ~20 features
  - All features are **categorical** (no numeric variables)

- **Balanced classes**: edible vs poisonous fairly even (≈50/50 split)
  - Some features show **strong correlation** with class (e.g., odor, spore print color)
  - Others appear less informative (e.g., veil color)



Data Readiness & Preprocessing

- **Can the data be fed directly? → No**
  - Features are **categorical**, not numeric
  - Some attributes are **redundant or low-value**
  - Dataset must be **transformed before modeling**
- **Planned Preprocessing Steps**
  - Encode categorical features (one-hot, label encoding)
  - Remove/merge redundant features (e.g., veil color)
  - Train-test split & cross-validation setup
  - Normalize if required for certain models (e.g., logistic regression)

Data Quality After Preprocessing

- **After Preprocessing**
  - Dataset fully numeric → suitable for ML models
  - Redundant/low-value features removed
  - Encoded features standardized into consistent format
  - No missing values → complete dataset
  - Balanced class distribution preserved
  - Ready for **training, validation, and testing**
  - Raw: odor=almond, cap-shape=convex
  - Encoded: [odor_almond=1, odor_fishy=0, cap_shape_convex=1, cap_shape_flat=0].)

Next Steps

- Implement baseline models (Logistic Regression, Decision Tree)
- Train advanced models (Random Forest, XGBoost, Neural Network)
- Compare results across models using **MCC** and other metrics
- Perform hyperparameter tuning & feature selection
- Document results for **progress checkpoints & final report**

Presentation link

- https://drive.google.com/file/d/1_OgVNoOaIpmptJeDjNmaBbNOz_smlJ6K/view?usp=sharing