# ARCHITECTURE

# Mushroom Classification

Revision Number: 1.5
Last date of revision: 25th October 2021

**Riyaz Khorasi & Sachin Shinde**

# Document Version Control

| Date Issued | Version | Description | Author |
|---|---|---|---|
| **25 Oct 2021** | 1.1 | First Draft | Sachin Shinde |
| **25 Oct 2021** | 1.2 | Added Workflow chart | Riyaz Khorasi |
| **25 Oct 2021** | 1.3 | Added user I/O flowchart | Sachin Shinde |
| **25 Oct 2021** | 1.4 | Added dataset overview and updated user I/O flowchart. | Sachin Shinde |
| **25 Oct 2021** | 1.5 | Restructure and reformat LLD | Riyaz Khorasi |

# Contents

# Abstract

In recent years, the popularity of mushrooms as a superfood and the understanding of their vast health benefits has surged. What used to be typically seen as just a traditional food, mushrooms are now being widely consumed and acknowledged for their healing and health abilities. There are thousands of species of Mushrooms in the world; they are edible and non-edible being poisonous. It is difficult for non-expertise people to Identify poisonous and edible mushrooms of all species manually. So, a computer aided system with software or algorithm is required to classify poisonous and nonpoisonous mushrooms. This project is presented on classification of poisonous and nonpoisonous mushrooms. Most of the research works to classify the type of mushroom have applied machine learning techniques like Naïve Bayes, K-Neural Network, Support vector Machine (SVM), Decision Tree techniques. In this task, a summary and comparisons of all different machine learning models of mushroom classification in terms of its performance parameters.

# 1 Introduction

## 1.1 Why this Low-Level Design Document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Mushroom Classification. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

## 1.2 Scope

This software system will be a Web application. This system will be designed to predict the classification of mushrooms based on the input by the user. Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

## 1.3 Constraints

The Mushroom Classification prediction application must be user friendly, as automated as possible and users should not be required to know any of the workings.

## 1.4 Risks

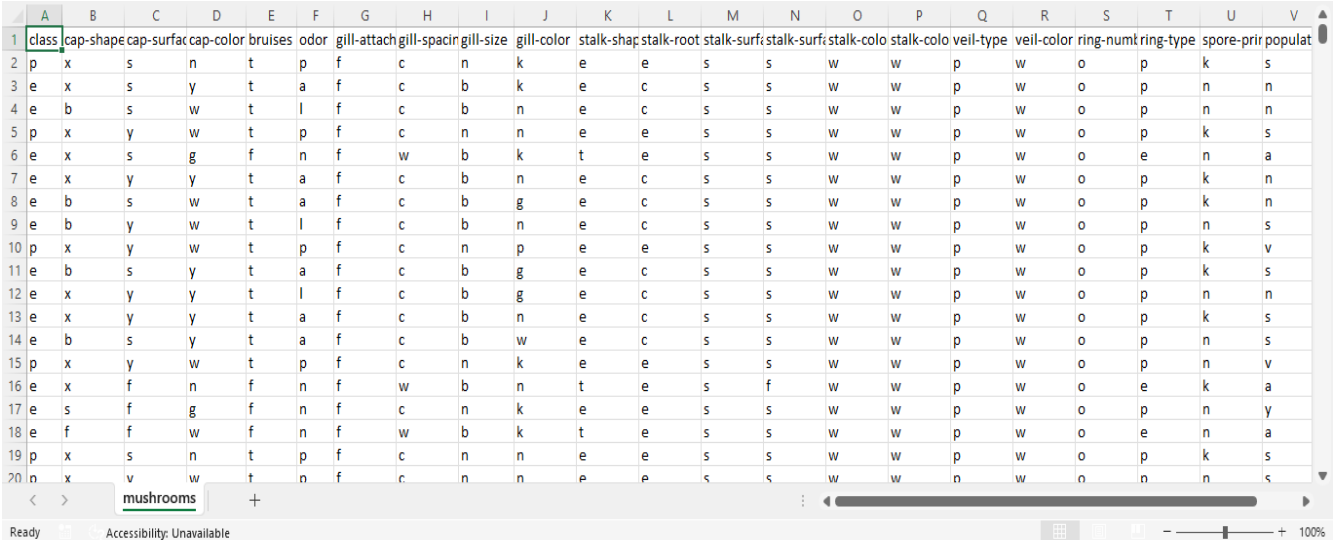
Document specific risks that have been identified or that should be considered.

## 1.5 Out of Scope

Delineate specific activities, capabilities, and items that are out of scope for the project.

# 2 Technical specifications

## 2.1 Dataset



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | class | cap-shape | cap-surfac | cap-color | bruises | odor | gill-attach | gill-spacir | gill-size | gill-color | stalk-shap | stalk-root | stalk-surfa | stalk-surfa | stalk-colo | stalk-colo | veil-type | veil-color | ring-numt | ring-type | spore-prir | populat |
| 2 | p | x | s | n | t | p | f | c | n | k | e | e | s | s | w | w | p | w | o | p | k | s |
| 3 | e | x | s | y | t | a | f | c | b | k | e | c | s | s | w | w | p | w | o | p | n | n |
| 4 | e | b | s | w | t | l | f | c | b | n | e | c | s | s | w | w | p | w | o | p | n | n |
| 5 | p | x | y | w | t | p | f | c | n | n | e | e | s | s | w | w | p | w | o | p | k | s |
| 6 | e | x | s | g | f | n | f | w | b | k | t | e | s | s | w | w | p | w | o | e | n | a |
| 7 | e | x | y | y | t | a | f | c | b | n | e | c | s | s | w | w | p | w | o | p | k | n |
| 8 | e | b | s | w | t | a | f | c | b | g | e | c | s | s | w | w | p | w | o | p | k | n |
| 9 | e | b | y | w | t | l | f | c | b | n | e | c | s | s | w | w | p | w | o | p | n | s |
| 10 | p | x | y | w | t | p | f | c | n | p | e | e | s | s | w | w | p | w | o | p | k | v |
| 11 | e | b | s | y | t | a | f | c | b | g | e | c | s | s | w | w | p | w | o | p | k | s |
| 12 | e | x | y | y | t | l | f | c | b | g | e | c | s | s | w | w | p | w | o | p | n | n |
| 13 | e | x | y | y | t | a | f | c | b | n | e | c | s | s | w | w | p | w | o | p | k | s |
| 14 | e | b | s | y | t | a | f | c | b | w | e | c | s | s | w | w | p | w | o | p | n | s |
| 15 | p | x | y | w | t | p | f | c | n | k | e | e | s | s | w | w | p | w | o | p | n | v |
| 16 | e | x | f | n | f | n | f | w | b | n | t | e | s | f | w | w | p | w | o | e | k | a |
| 17 | e | s | f | g | f | n | f | c | n | k | e | e | s | s | w | w | p | w | o | p | n | y |
| 18 | e | f | f | w | f | n | f | w | b | k | t | e | s | s | w | w | p | w | o | e | n | a |
| 19 | p | x | s | n | t | p | f | c | n | n | e | e | s | s | w | w | p | w | o | p | k | s |
| 20 | n | x | y | w | t | n | f | c | n | n | e | e | s | s | w | w | p | w | o | p | n | s |

mushrooms

Ready Accessibility: Unavailable 100%

### 2.1.1 Dataset overview

The dataset consists of a table with 8,124 records and 23 features.

Attribute Information: (classes: edible=e, poisonous=p)

- cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
- cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
- cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y
- bruises: bruises=t,no=f
- odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
- gill-attachment: attached=a,descending=d,free=f,notched=n
- gill-spacing: close=c,crowded=w,distant=d
- gill-size: broad=b,narrow=n
- gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y
- stalk-shape: enlarging=e,tapering=t
- stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?
- stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- veil-type: partial=p,universal=u
- veil-color: brown=n,orange=o,white=w,yellow=y
- ring-number: none=n,one=o,two=t
- ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
- spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y

- population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
- habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

### 2.1.2 Input schema

| S.no | Feature Name | Required/Null |
|---|---|---|
| 1. | cap-shape | Required |
| 2. | cap-surface | Required |
| 3. | cap-color | Required |
| 4. | bruises | Required |
| 5. | Odor | Required |
| 6. | gill-spacing | Required |
| 7. | gill-size | Required |
| 8. | gill-color | Required |
| 9. | stalk-shape | Required |
| 10. | stalk-root | Required |
| 11. | stalk-surface-above-ring | Required |
| 12. | stalk-surface-below-ring | Required |
| 13. | stalk-color-above-ring | Required |
| 14. | stalk-color-below-ring | Required |
| 15. | ring-number | Required |
| 16. | ring-type | Required |
| 17. | spore-print-color | Required |
| 18. | population | Required |
| 19. | habitat | Required |

## 2.2 Predicting Classification

- The system presents the set of inputs required from the user.
- The user gives required information.
- The system then predicts classification of given the above inputs.

## 2.3 Logging

We should be able to log every activity done by the user.

- The System identifies at what step logging required
- The System should be able to log each and every system flow.
- Developers can choose logging methods. You can choose database logging/ File logging as well.
- System should not be hung even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

## 2.4 Deployment

✓ HEROKU

# 3 Technology stack

| | |
|---|---|
| **Front End** | HTML/CSS |
| **Backend** | Python Flask |
| **Deployment** | Heroku |

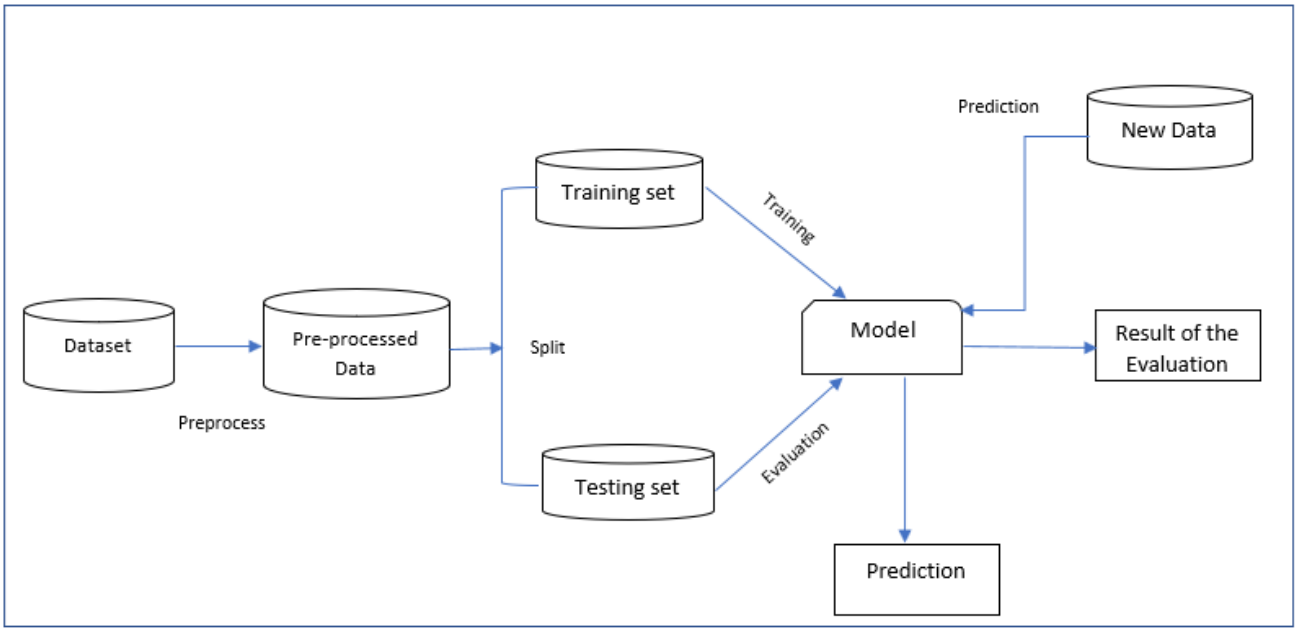
# 4 Proposed Solution

The proposed solution for this project is Machine learning algorithms can be implemented to predict the classification of mushrooms. Considering various features as inputs from the web app, the implemented classification model will predict the output as mushrooms are Edible or Poisonous. Here, we have used Random Forest Classifier to predict and classify whether the mushrooms are Edible or Poisonous.

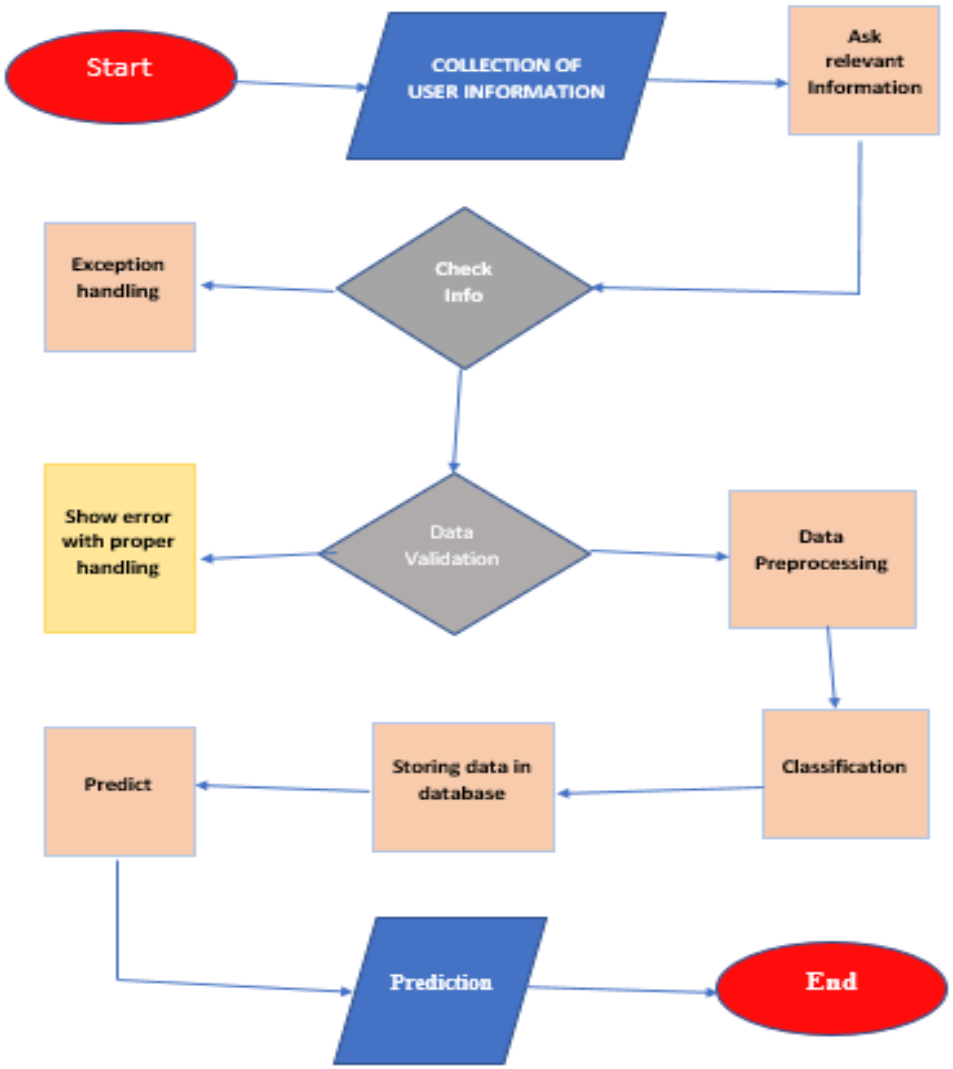
However, drawing a baseline model is important since it tells us how well other models have performed compared to base model. Here, the base model for Mushroom Classification is Logistic Regression.

1. Baseline Model : Logistic Regression
2. Actual Model : Random Forest

# 5 Model training/validation workflow



# 6 User I/O workflow



# 7 Exceptional scenarios

| Step | Exception | Mitigation | Module |
|---|---|---|---|
| 25 Oct 2021 | 1.1 | First Draft | **Sachin Shinde** |
| 25 Oct 2021 | 1.2 | Added Workflow chart | **Riyaz Khorasi** |

# 8 Test cases

| Test case | Steps to perform test case | Module | Pass/Fail |
|---|---|---|---|
| 1 | Checking model execution and Evaluation | ML modelling | Pass |
| 2 | User Input and Prediction test | Deployment case | Pass |

# 9 Performance

We can observe that the accuracy of the predicted output was seen at 99% using Random Forest classifier. Other classification models such as logistic regression and decision tree have given good accuracy above 97% and 83% respectively.