# DPR

**MUSHROOM CLASSIFICATION**

***PARAS DAHIYA***

**ABSTRACT**

Mushrooms can be found extensively in a variety of natural environments and visual identification of mushroom species is well established. Some mushrooms are known because of their nutritional and therapeutical properties. Some species are known all over the world because of their toxicity that causes fatal accidents every year mainly due to misidentification. Some of the edible mushrooms are Ganoderma spp, Cantharellus spp, Agaricus spp, Pleurotus spp, Russula spp, Auricularia spp and Termitomyces spp; but the ornamentals are the beautifully ringed Microporous spp. Amanita spp, Lepiota cristata, Lepiota brunneoincarnata and Inocybe asterospora, C o p r i n u s s p p are among the most important species responsible for mushroom poisoning. Morphological and chemical analyses for mushrooms are occasionally required in forensic science practice. In this work, we will try to predict which mushroom is poisonous & which is edible.

**CONTENTS**

Title Page i

Abstract ii

Content iii

|  |  |  |
| --- | --- | --- |
|  | Introduction | 1 |
|  | Why this DPR Documentation? | 1 |
|  | Key points: | 1 |
| 1 | Description | 1 |
| 1.1 | Problem Perspective | 1 |
| 1.2 | Problem Statement | 1 |
| 1.3 | Proposed Solution | 1 |
| 2 | Technical Requirements | 2 |
| 2.1 | Tools Used | 2 |
| 3 | Data Requirements | 3 |
| 3.1 | Data Gathering from Main Source | 3 |
| 3.2 | Data Description | 3 |
| 3.3 | Import Data into Database | 3 |
| 3.4 | Export Data from Database | 4 |
| 4 | Data Pre-Processing | 4 |
| 5 | Design Flow | 4 |
| 5.1 | Modelling | 4 |
| 5.2 | UI Integration | 4 |
| 5.3 | Modelling Process | 5 |
| 5.4 | Deployment Process | 5 |
| 6 | Data Validation | 6 |
| 7 | Model Training | 7 |
| 8 | Prediction | 8 |
| 9 | Deployment | 8 |
| 10 | Conclusion | 8 |
|  |  |  |

# INTRODUCTION

## Why this DPR Documentation?

The main purpose of this DPR documentation is to add the necessary details of the project and provide the description of the machine learning model and the written code. This also provides the detailed description on how the entire project has been designed end-to-end.

### Key points:

* Describes the design flow
* Implementations
* Software requirements
* Architecture of the project
* Non-functional attributes like:
  + Reusability
  + Portability
  + Resource utilization

# 1 Description

## Problem Perspective

Mushroom classifier is a machine learning model which helps us to predict wheter the mushroom is edible or poisionous.

## Problem Statement

The Audubon Society Field Guide to North American Mushrooms contains descriptions

of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom (1981). Each species is labelled as either definitely edible, definitely poisonous, or maybe edible but not recommended. This last category was merged with the toxic category. The Guide asserts unequivocally that there is no simple rule for judging a mushroom's edibility, such as "leaflets three, leave it be" for Poisonous Oak and Ivy.

The main goal is to predict which mushroom is poisonous & which is edible.

## Proposed Solution

The solution proposed to take dataset from the client and process all the provided data to meet the requirements of the machine learning model and finally displaying the output csv file.

# Technical Requirements

There are no hardware requirements required for using this application, the user must have an interactive device which has access to the internet and must have the basic understanding of providing the input. And for the backend part the server must run all the software that is required for the processing the provided data and to display the results.

## Tools Used

* + - Python 3.6 is used as the programming language and frame works like numpy, pandas, sklearn and other modules for building the model.
    - PyCharm is used as IDE.
    - For visualizations seaborn and parts of matplotlib are being used.
    - For data collection SQl database is being used.
    - Front end development is done using HTML/CSS/JAVASCRIPT.
    - Flask is used for both data and backend deployment.
    - GitHub is used for version control.
    - Heroku is used for deployment.

# 3. Data Requirements

The data requirement is completely based on the problem statement. And the data set is available on the Kaggle in the form of excel sheet(.xlsx). As the main theme of the project is to get the experience of real time problems, we are again importing the data into the SQL data base and exporting it into csv format.

## 3.1 Data Gathering from Main Source

The data for the current project is being gathered from Kaggle dataset, the link to the data is:

https://www.kaggle.com/datasets/uciml/mushroom-classification

## 3.2 Data Description

Data Description: This dataset describes mushrooms in terms of their physical characteristics. They are classified into: poisonous or edible.

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s

2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s

3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y

4. bruises?: bruises=t,no=f

5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s

6. gill-attachment: attached=a,descending=d,free=f,notched=n

7. gill-spacing: close=c,crowded=w,distant=d

8. gill-size: broad=b,narrow=n

9. 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

10. stalk-shape: enlarging=e,tapering=t

11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?

12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s

13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s

14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y

15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y

16. veil-type: partial=p,universal=u

17. veil-color: brown=n,orange=o,white=w,yellow=y

18. ring-number: none=n,one=o,two=t

19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z

20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y

21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y

22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

## 3.3 Import Data into Database

Created an api for the upload of the data into the SQlL database, steps performed are:

* + - Connection is made with the database.
    - Created a database with name Prediction.
    - SQL command is written for creating the data table with required parameters.
    - And finally, a SQL command is written for uploading the dataset into the data table by bulk insertion.

## 3.4 Export Data from Database

In the above created api, the download url is also being created, which downloads the data into a csv file format.

# 4. Data Pre-Processing

Steps performed in pre-processing are:

First the data types are being checked

There was no missing vlaues in the data. But when we go through the data, we find that missing values in one column is replaced with '?'.

Replaced such values with numpy "nan" so that we can handle the missing values.

Performed categorical imputing for the required columns.

Added quote on the the whole data.

And, the data is ready for passing to the machine learning algorithm.

# 5. Design Flow

**5.1 Modeling**

The pre-processed data is then visualized and all the required insights are being drawn. Although from the drawn insights, the data is randomly spread but still modeling is performed with different machine learning algorithms to make sure we cover all the possibilities. And finally, as expected random forest regression performed well and further hyperparameter tuning is done to increase the model’s accuracy.

## 5.2 UI Integration

Both CSS, HTML and JAVASCRIPT files are being created and are being integrated with the created machine learning model. All the required files are then integrated to the app.py file and tested locally.

**5.3 Modelling Process&5.4 Deployment Process**



# 6. Data Validation

In this step, we perform different sets of validation on the given set of training files.

1. Name Validation- We validate the name of the files based on the given name in the schema file. We have created a regex pattern as per the name given in the schema file to use for validation. After validating the pattern in the name, we check for the length of date in the file name as well as the length of time in the file name. If all the values are as per requirement, we move such files to "Good\_Data\_Folder" else we move such files to "Bad\_Data\_Folder."
2. Number of Columns - We validate the number of columns present in the files, and if it doesn't match with the value given in the schema file, then the file is moved to "Bad\_Data\_Folder."
3. Name of Columns - The name of the columns is validated and should be the same as given in the schema file. If not, then the file is moved to "Bad\_Data\_Folder".
4. The datatype of columns - The datatype of columns is given in the schema file. This is validated when we insert the files into Database. If the datatype is wrong, then the file is moved to "Bad\_Data\_Folder".
5. Null values in columns - If any of the columns in a file have all the values as NULL or missing, we discard such a file and move it to "Bad\_Data\_Folder".

# 7. Model Training

1) Data Export from Db - The data in a stored database is exported as a CSV file to be used for model training.

2) Data Preprocessing

a) Check for null values in the columns. If present, impute the null values using the KNN imputer.

b) Check if any column has zero standard deviation, remove such columns as they don't give any information during model training.

3) Clustering - KMeans algorithm is used to create clusters in the preprocessed data. The optimum number of clusters is selected by plotting the elbow plot, and for the dynamic selection of the number of clusters, we are using "KneeLocator" function. The idea behind clustering is to implement different algorithms

To train data in different clusters. The Kmeans model is trained over preprocessed data and the model is saved for further use in prediction.

4) Model Selection - After clusters are created, we find the best model for each cluster. We are using two algorithms, "Random Forest" and "KNN". For each cluster, both the algorithms are passed with the best parameters derived from GridSearch. We calculate the AUC scores for both models and select the model with the best score. Similarly, the model is selected for each cluster. All the models for every cluster are saved for use in prediction.

# 8. Prediction

1) Data Export from Db - The data in the stored database is exported as a CSV file to be used for prediction.

2) Data Preprocessing

a) Check for null values in the columns. If present, impute the null values using the KNN imputer.

b) Check if any column has zero standard deviation, remove such columns as we did in training.

3) Clustering - KMeans model created during training is loaded, and clusters for the preprocessed prediction data is predicted.

4) Prediction - Based on the cluster number, the respective model is loaded and is used to predict the data for that cluster.

5) Once the prediction is made for all the clusters, the predictions along with the Wafer names are saved in a CSV file at a given location and the location is returned to the client.

# 9. Deployment

The tested model is then deployed to AWS. So, users can access the project from any internet devices.

# 10. Conclusion

The Mushroom Classifier can predict whether the mushroom is edible or poisionous based on the trained data set in the algorithm.