Mushroom Classification

Problem Statement

The goal is to build a machine learning model which predicts whether a mushroom is edible or poisonous based on the characteristics. There are two classes: 'e' and 'p'.

- 'e' means that the mushroom is edible.
- 'p' means that the mushroom is poisonous.

Data Description

This dataset describes mushrooms in terms of their physical characteristics.

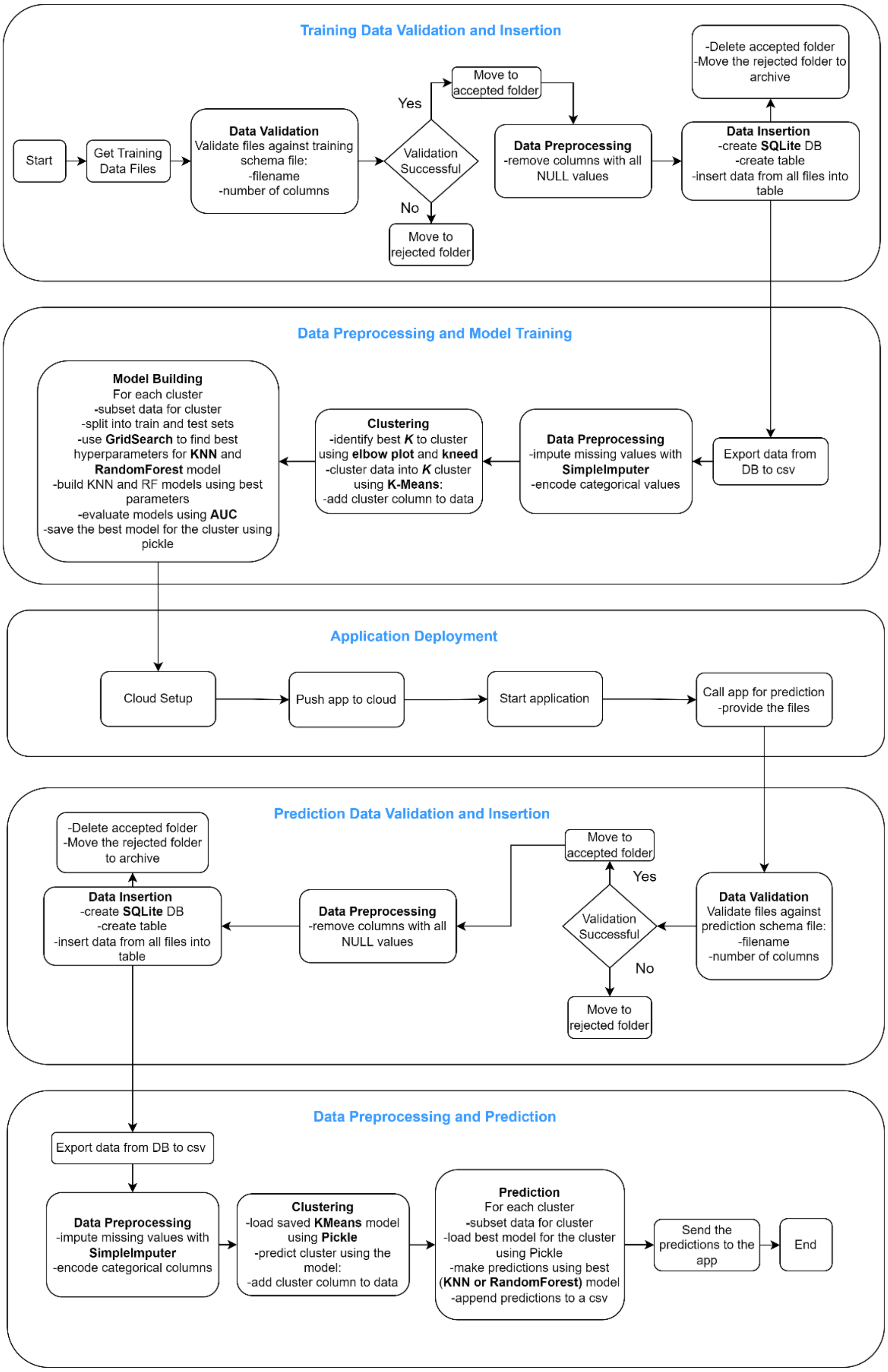
1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
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

Data is available as multiple sets of files. Each file will contain physical characteristics and a column to indicate whether it is edible [‘e’] or not [‘p']. Apart from data files, schema files are provided as a part of Data Sharing Agreement which contains all the relevant information about both train and test data such as:

- File name convention
- No of columns in each file
- Data type of each column
- Name of the columns

Architecture

Application Workflow


Training Data Validation and Insertion
Start
Get Training Data Files
Data Validation Validate files against training schema file: -filename -number of columns
Validation Successful
Yes
No
Move to accepted folder
Move to rejected folder
Data Preprocessing -remove columns with all NULL values
Data Insertion -create SQLite DB -create table -insert data from all files into table
-Delete accepted folder -Move the rejected folder to archive
Data Preprocessing and Model Training
Export data from DB to csv
Data Preprocessing -impute missing values with SimpleImputer -encode categorical values
Clustering -identify best K to cluster using elbow plot and kneed -cluster data into K cluster using K-Means: -add cluster column to data
Model Building For each cluster -subset data for cluster -split into train and test sets -use GridSearch to find best hyperparameters for KNN and RandomForest model -build KNN and RF models using best parameters -evaluate models using AUC -save the best model for the cluster using pickle
Application Deployment
Cloud Setup
Push app to cloud
Start application
Call app for prediction -provide the files
Prediction Data Validation and Insertion
Data Validation Validate files against prediction schema file: -filename -number of columns
Validation Successful
Yes
No
Move to accepted folder
Move to rejected folder
Data Preprocessing -remove columns with all NULL values
Data Insertion -create SQLite DB -create table -insert data from all files into table
-Delete accepted folder -Move the rejected folder to archive
Data Preprocessing and Prediction
Export data from DB to csv
Data Preprocessing -impute missing values with SimpleImputer -encode categorical columns
Clustering -load saved KMeans model using Pickle -predict cluster using the model: -add cluster column to data
Prediction For each cluster -subset data for cluster -load best model for the cluster using Pickle -make predictions using best (KNN or RandomForest) model -append predictions to a csv
Send the predictions to the app
End

Data Validation

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

- 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 "accepted" folder else we move such files to "rejected" folder.

- 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 "rejected" folder.

- 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 "rejected" folder.

- 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 "rejected" folder.

- Null values in Columns - If any of the columns in a file have all the values as NULL or missing, we discard such file and move it to "rejected" folder.

Data Insertion in Database

- Database Creation and connection - Create a database with the given name passed. If the database is already created, open the connection to the database.

- Table creation in the database - Table with name - "accepted", is created in the database for inserting the data of the files in the "accepted" folder. If the table is already present, then the new table is not created and new files are inserted in the already present table as we want training to be done on new as well as old training files.

- Insertion of files in the table - All the files in the "accepted" folder are inserted in the above created table. If any file has invalid data type in any of the columns, the file is not loaded in the table and is moved to "rejected" folder.

Model Training

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

- Data Pre-processing
    - Check for null values in the columns. If present, impute the null values using the Simple Imputer.
    - Encode categorical variables.

- Clustering - KMeans algorithm is used to create clusters in the pre-processed 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 "Knee Locator" function. The idea behind clustering is to implement different algorithms to train data in different clusters. Model is saved for further use in prediction.

- 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.

Prediction

- Similar to the training data, we perform data validation using test schema and then insert the data into SQLite database.

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

- Data Pre-processing
    - Check for null values in the columns. If present, impute the null values using the Simple Imputer.
    - Encode categorical variables.

- Clustering - KMeans model created during training is loaded, and clusters for the pre-processed prediction data is predicted.

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

- Once the prediction is made for all the clusters, the predictions along with the Wafer names are saved in a CSV file and is returned to the application.