Architecture Document

Mushroom Type Classifier

Revision No: 1.0

Last Date of Revision: 26/05/2023.

Document Version Control-

Date Issued	Version	Description	Author
26/05/2023	1.0	LLD – V- 1.0	Mahesh. A

Contents -

Description	Page No.
Document Version Control	2
1. Introduction	4
1.1 What is Low-Level design document	4
1.2 Scope	4
2. Architecture	4
3. Architecture Description	5
3.1 Data Accessing	5
3.2 Data Pre-Processing	7
3.3 Splitting the Data	9
3.4 Model Building	9
3.5 Create Front End User Module using flask	9
3.6 Testing the Model	12
4.0 KPI	17

Abstract

By using the pre-trained of Mushroom Classifier user can understand the type of mushroom he is handling by this we can reduce the risk involved in the intake of the same.

1 Introduction

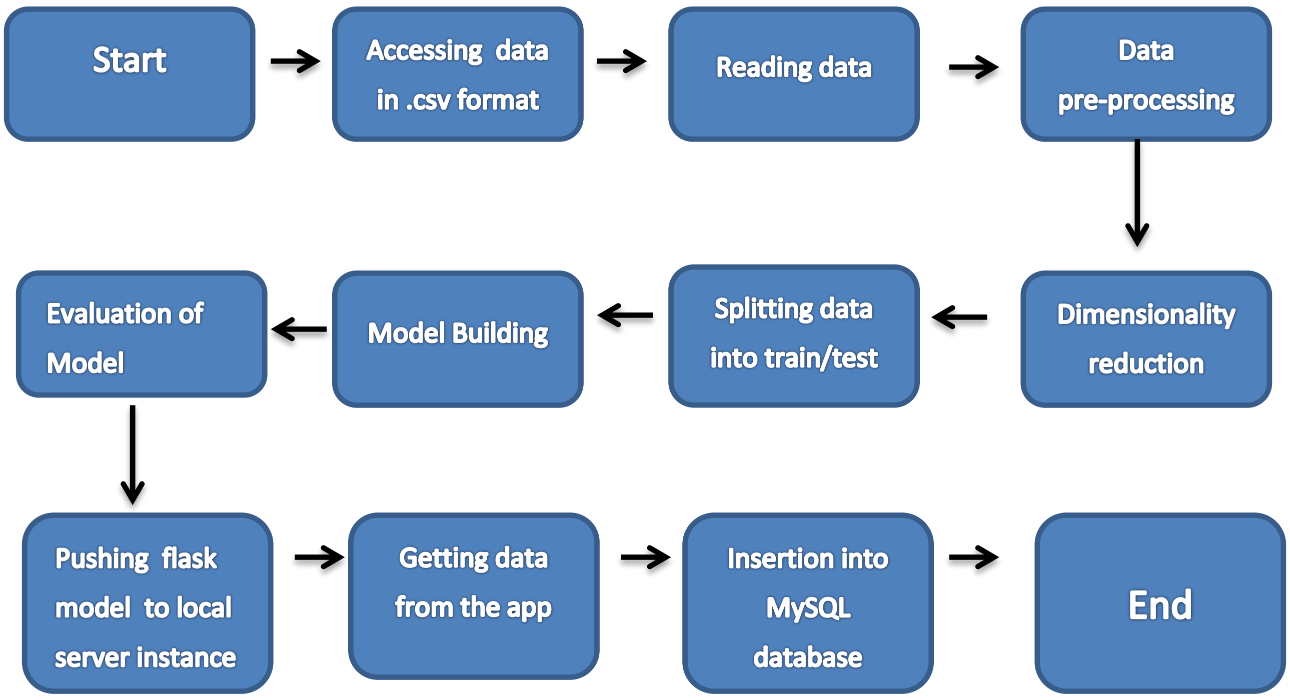
1.1. What is document required?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual programmed code for Mushroom classifier. LLD describes the class relations with predictors .It describes the modules so that the programmer can directly code the program from the document.

1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step r 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.

2 Architecture


Start
Accessing data in .csv format
Reading data
Data pre-processing
Dimensionality reduction
Splitting data into train/test
Model Building
Evaluation of Model
Pushing flask model to local server instance
Getting data from the app
Insertion into MySQL database
End

3 Architecture Description

3.1 Data Accessing

We can access the data in the from the download link as available in the project.

We load the data to the framework using the pandas read function.

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

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

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

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

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

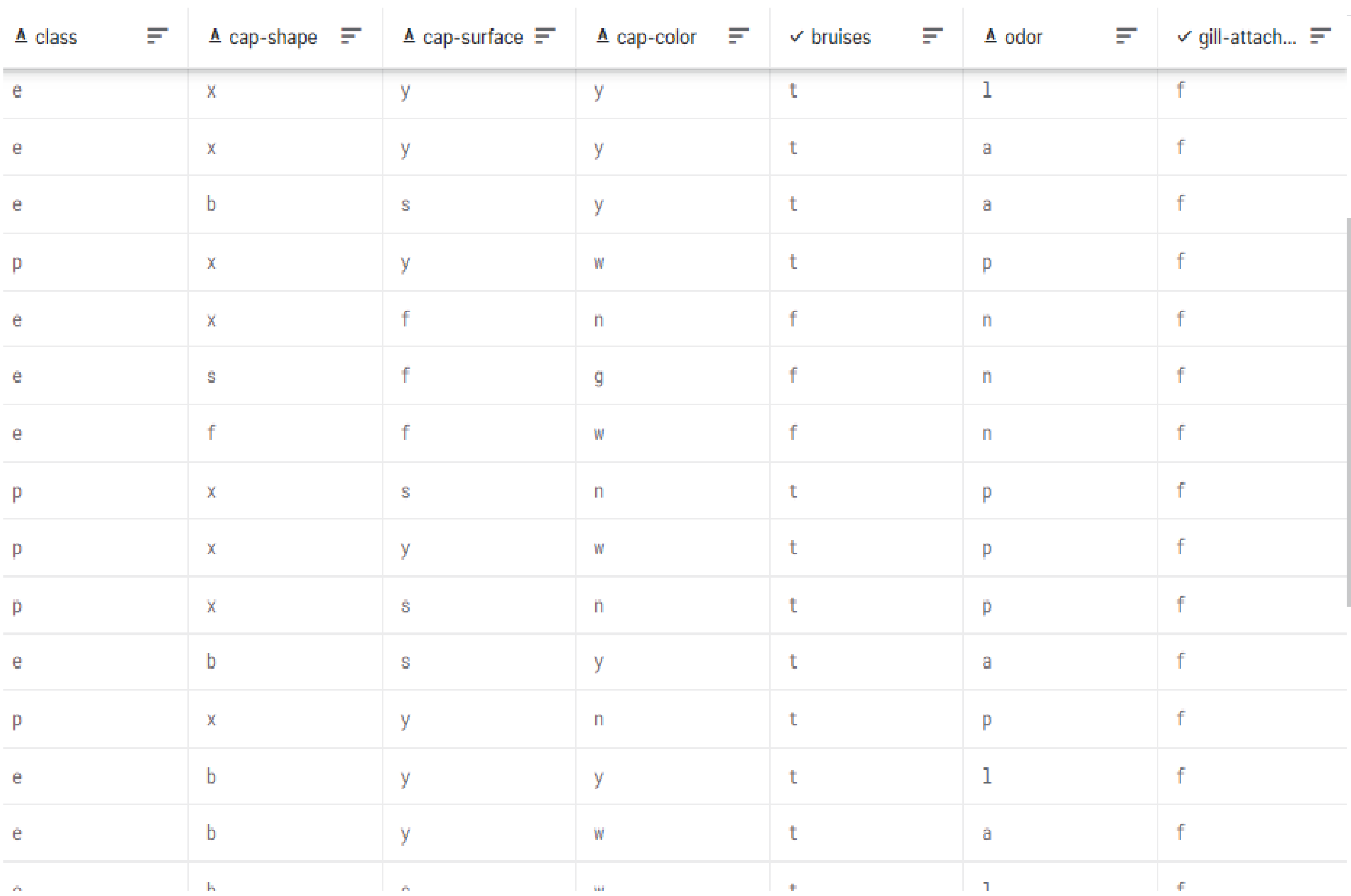
r. ring-number: none=n, one=o, two=t

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

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

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

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

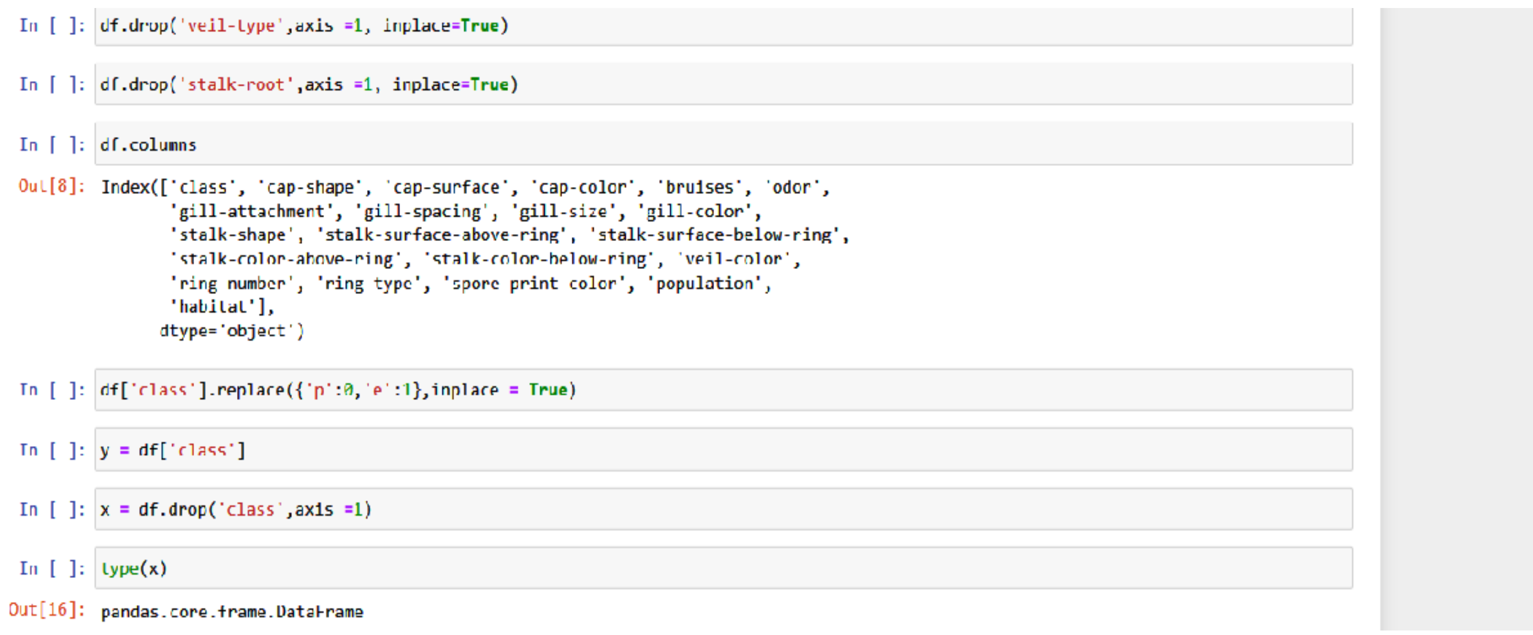


class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attach...
e	x	y	y	t	l	f
e	x	y	y	t	a	f
e	b	s	y	t	a	f
p	x	y	w	t	p	f
e	x	f	n	f	n	f
e	s	f	g	f	n	f
e	f	f	w	f	n	f
p	x	s	n	t	p	f
p	x	y	w	t	p	f
p	x	s	n	t	p	f
e	b	s	y	t	a	f
p	x	y	n	t	p	f
e	b	y	y	t	l	f
e	b	y	w	t	a	f

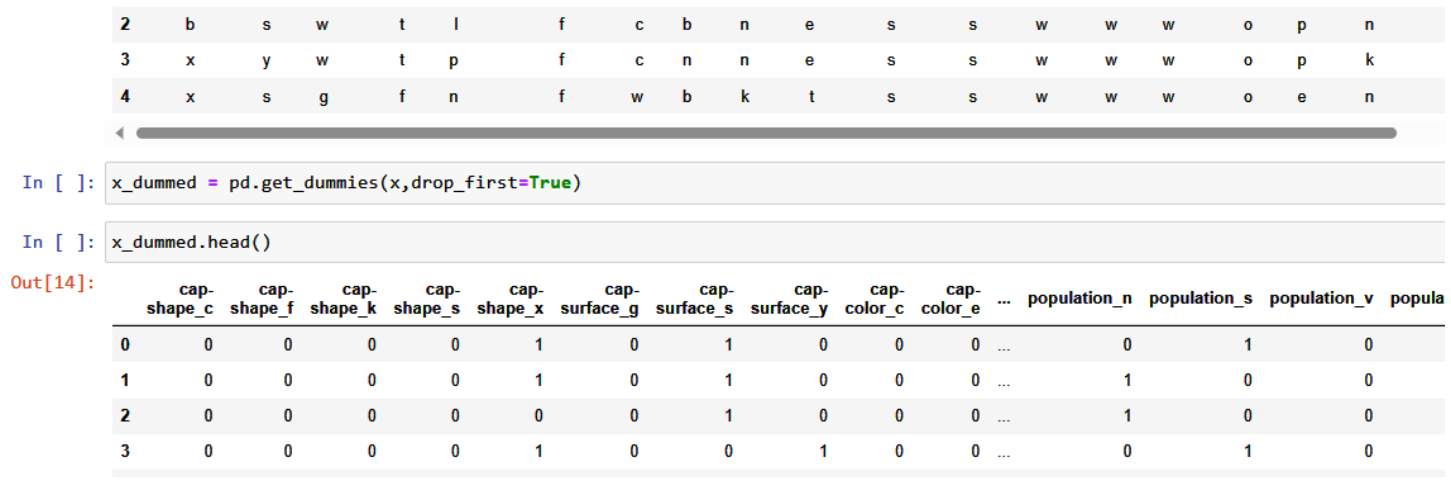
3.2 Data Pre-Processing

By the usage of the different data manipulation techniques we will remove unwanted features

Also we use the dimensionality reduction and make all the features in numerical data type



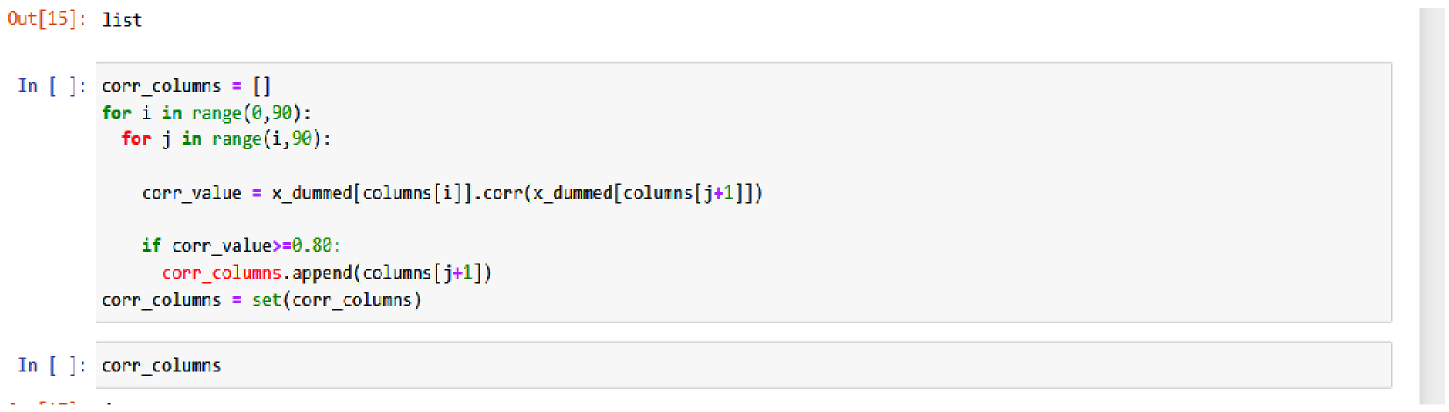
```
In [ ]: df.drop('veil-type',axis =1, inplace=True)

In [ ]: df.drop('stalk-root',axis =1, inplace=True)

In [ ]: df.columns

Out[8]: Index(['class', 'cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor',
               'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color',
               'stalk-shape', 'stalk-surface-above-ring', 'stalk-surface-below-ring',
               'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-color',
               'ring number', 'ring type', 'spore print color', 'population',
               'habitat'],
              dtype='object')

In [ ]: df['class'].replace({'p':0,'e':1},inplace = True)

In [ ]: y = df['class']

In [ ]: x = df.drop('class',axis =1)

In [ ]: type(x)

Out[16]: pandas.core.frame.DataFrame
```



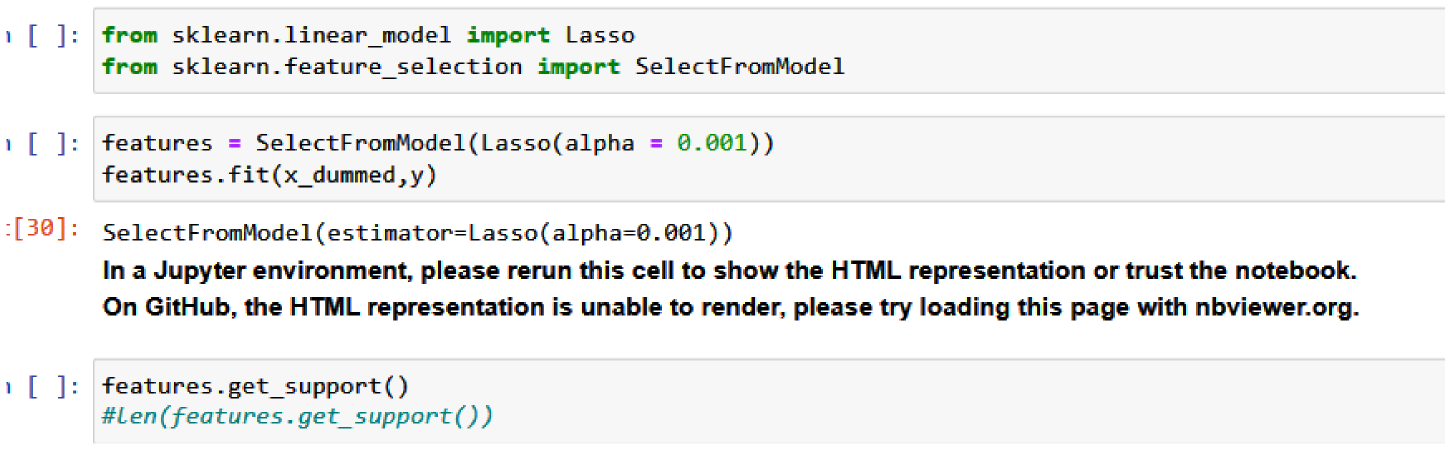
2	b	s	w	t	l	f	c	b	n	e	s	s	w	w	w	o	p	n
3	x	y	w	t	p	f	c	n	n	e	s	s	w	w	w	o	p	k
4	x	s	g	f	n	f	w	b	k	t	s	s	w	w	w	o	e	n

```
In [ ]: x_dummed = pd.get_dummies(x,drop_first=True)

In [ ]: x_dummed.head()

Out[14]:
```

	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_g	cap-surface_s	cap-surface_y	cap-color_c	cap-color_e	...	population_n	population_s	population_v	popula
0	0	0	0	0	1	0	1	0	0	0	...	0	1	0	
1	0	0	0	0	1	0	1	0	0	0	...	1	0	0	
2	0	0	0	0	0	0	1	0	0	0	...	1	0	0	
3	0	0	0	0	1	0	0	1	0	0	...	0	1	0	



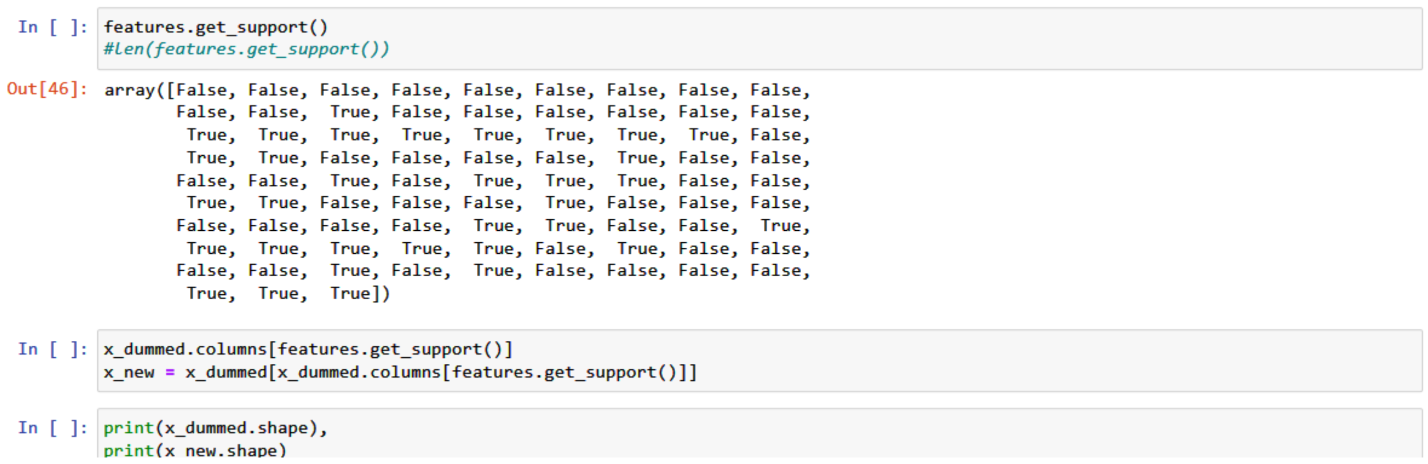
```
Out[15]: list

In [ ]: corr_columns = []
        for i in range(0,90):
          for j in range(i,90):

            corr_value = x_dummed[columns[i]].corr(x_dummed[columns[j+1]])

            if corr_value>=0.80:
              corr_columns.append(columns[j+1])
        corr_columns = set(corr_columns)

In [ ]: corr_columns
```



```
 [ ]: from sklearn.linear_model import Lasso
      from sklearn.feature_selection import SelectFromModel

 [ ]: features = SelectFromModel(Lasso(alpha = 0.001))
      features.fit(x_dummed,y)

:[30]: SelectFromModel(estimator=Lasso(alpha=0.001))
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

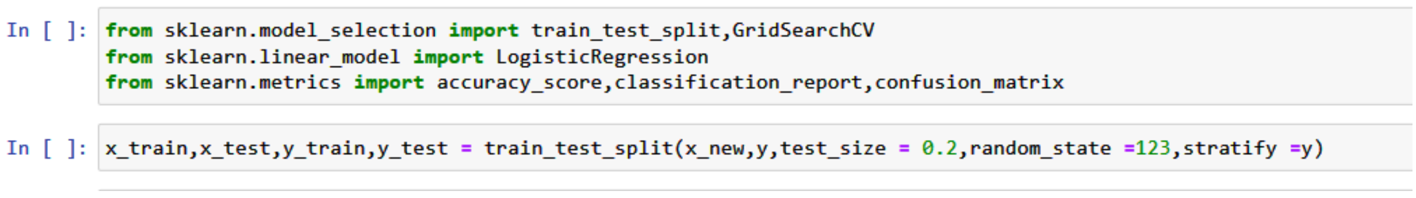
```
 [ ]: features.get_support()
      #len(features.get_support())
```



```
In [ ]: features.get_support()
        #len(features.get_support())

Out[46]: array([False, False, False, False, False, False, False, False, False,
                False, False,  True, False, False, False, False, False, False,
                 True,  True,  True,  True,  True,  True,  True,  True, False,
                 True,  True, False, False, False, False,  True, False, False,
                False, False,  True, False,  True,  True,  True, False, False,
                 True,  True, False, False, False,  True, False, False, False,
                False, False, False, False,  True,  True, False, False,  True,
                 True,  True,  True,  True,  True, False,  True, False, False,
                False, False,  True, False,  True, False, False, False, False,
                 True,  True,  True])

In [ ]: x_dummed.columns[features.get_support()]
        x_new = x_dummed[x_dummed.columns[features.get_support()]]

In [ ]: print(x_dummed.shape),
        print(x_new.shape)
```

3.3 Splitting the Data

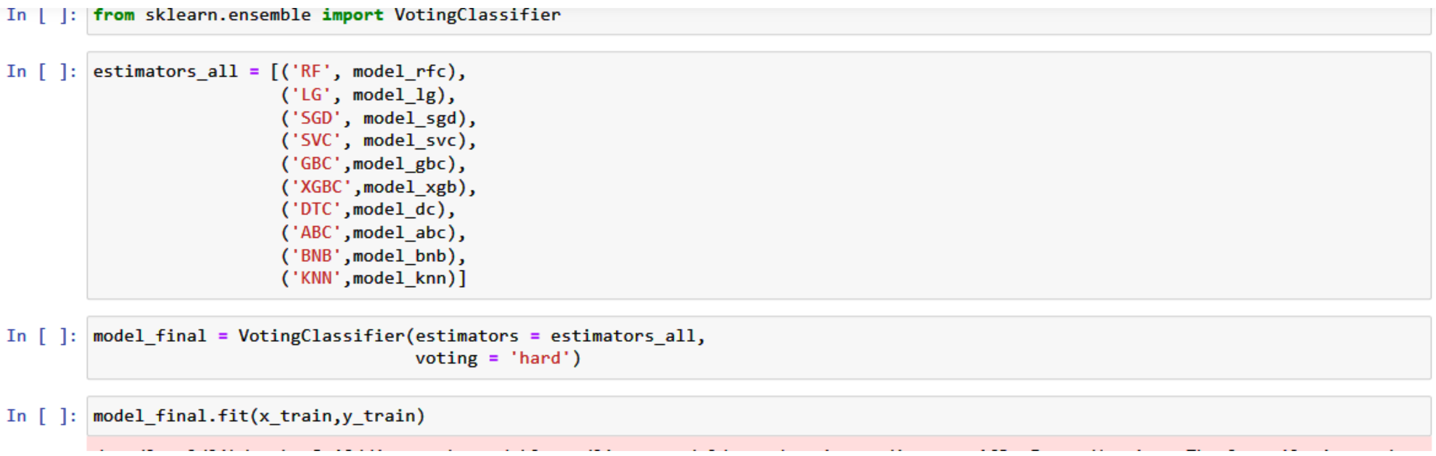
We use train test split Sklearn function to split the data for training and validation



```
In [ ]: from sklearn.model_selection import train_test_split,GridSearchCV
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

In [ ]: x_train,x_test,y_train,y_test = train_test_split(x_new,y,test_size = 0.2,random_state =123,stratify =y)
```

3.4 Model Building

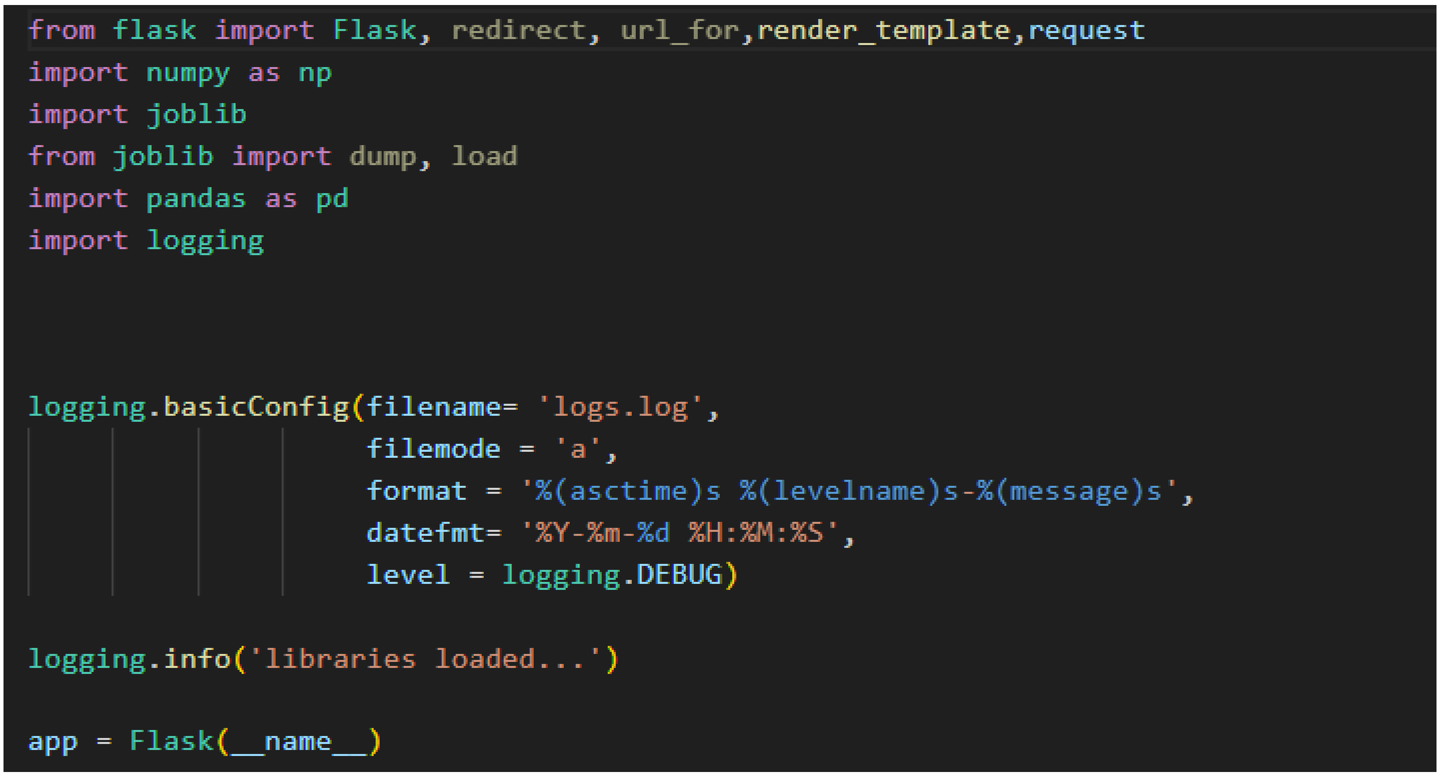
We will deploy as many models as possible and fine tune them using GridsearchCV select the model which are performing with highest accuracy and select those models and ensemble them using voting classifier for the best model. Perform the model evaluation.



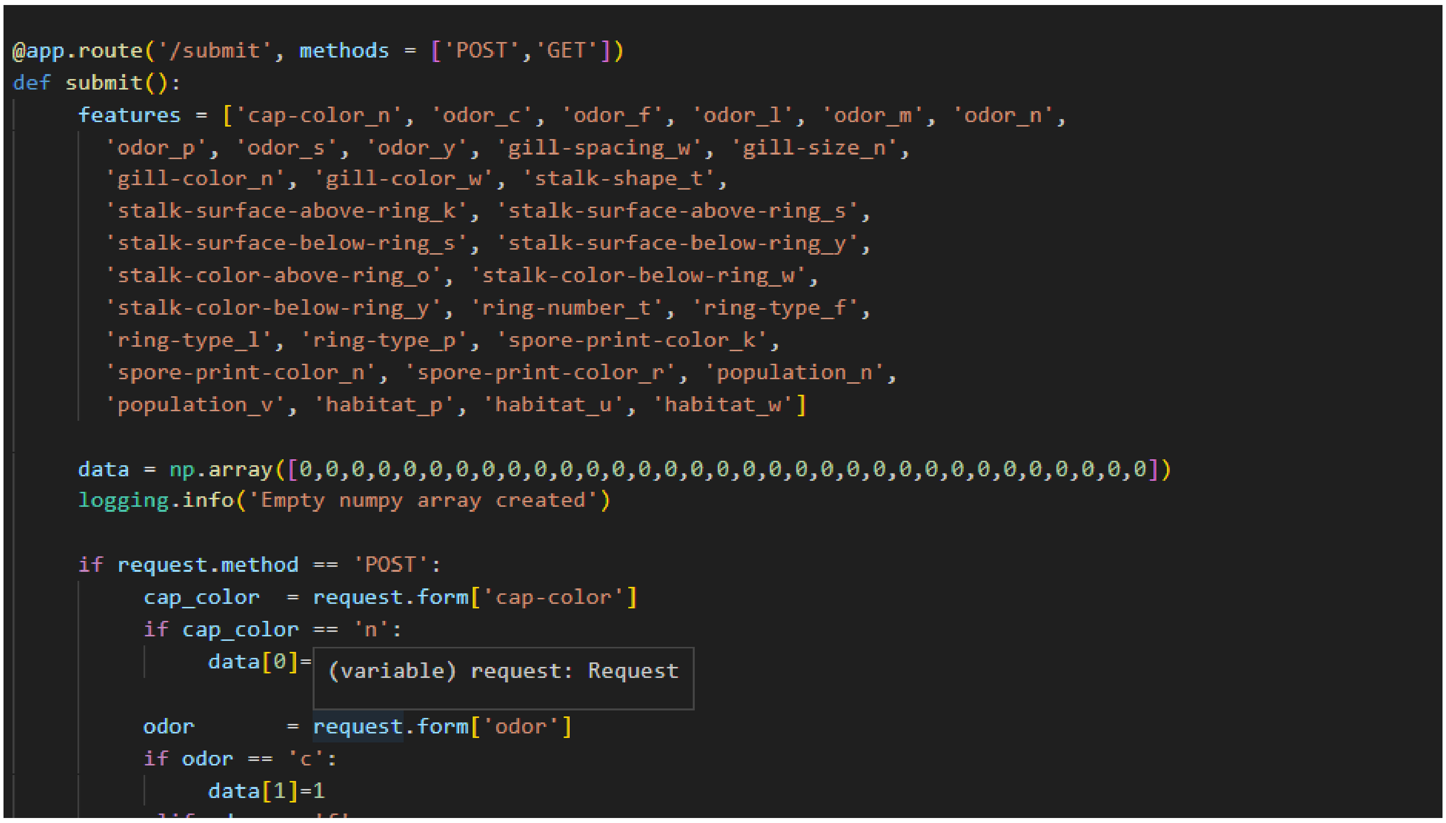
```
In [ ]: from sklearn.ensemble import VotingClassifier

In [ ]: estimators_all = [('RF', model_rfc),
                          ('LG', model_lg),
                          ('SGD', model_sgd),
                          ('SVC', model_svc),
                          ('GBC',model_gbc),
                          ('XGBC',model_xgb),
                          ('DTC',model_dc),
                          ('ABC',model_abc),
                          ('BNB',model_bnb),
                          ('KNN',model_knn)]

In [ ]: model_final = VotingClassifier(estimators = estimators_all,
                                       voting = 'hard')

In [ ]: model_final.fit(x_train,y_train)
```

3.5 Create Front End User Module using flask

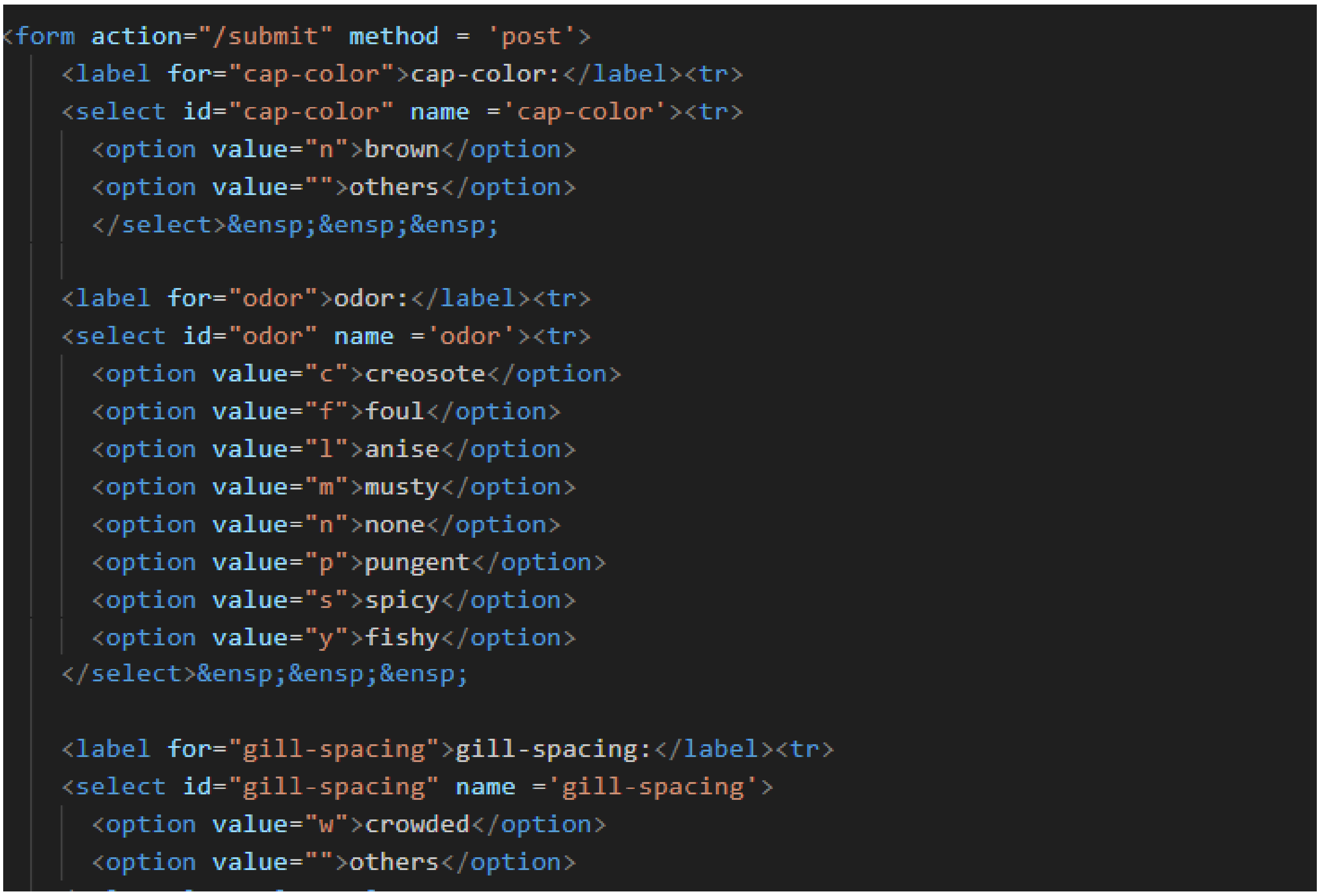
Once the model is created download and save the model and now we create GUI for front end user using the flask incorporated with HTML, CSS. Align and map the user data to the data base created. From user data create the data frame and load it to the model for the prediction the same prediction is send back to the user GUI and well saved in the data base (MySQL).



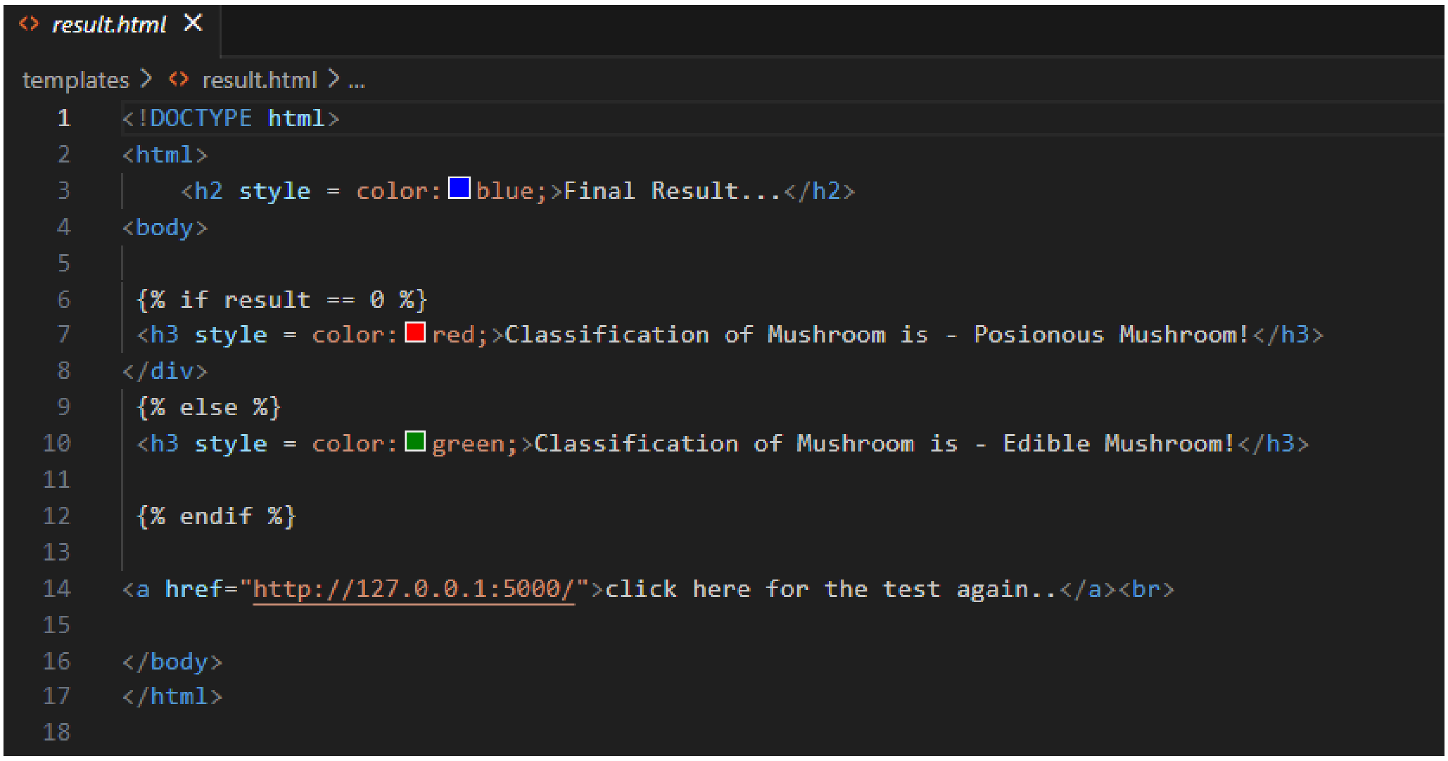
```python
from flask import Flask, redirect, url_for,render_template,request
import numpy as np
import joblib
from joblib import dump, load
import pandas as pd
import logging



logging.basicConfig(filename= 'logs.log',
                    filemode = 'a',
                    format = '%(asctime)s %(levelname)s-%(message)s',
                    datefmt= '%Y-%m-%d %H:%M:%S',
                    level = logging.DEBUG)

logging.info('libraries loaded...')

app = Flask(__name__)
```



```python
@app.route('/submit', methods = ['POST','GET'])
def submit():
    features = ['cap-color_n', 'odor_c', 'odor_f', 'odor_l', 'odor_m', 'odor_n',
      'odor_p', 'odor_s', 'odor_y', 'gill-spacing_w', 'gill-size_n',
      'gill-color_n', 'gill-color_w', 'stalk-shape_t',
      'stalk-surface-above-ring_k', 'stalk-surface-above-ring_s',
      'stalk-surface-below-ring_s', 'stalk-surface-below-ring_y',
      'stalk-color-above-ring_o', 'stalk-color-below-ring_w',
      'stalk-color-below-ring_y', 'ring-number_t', 'ring-type_f',
      'ring-type_l', 'ring-type_p', 'spore-print-color_k',
      'spore-print-color_n', 'spore-print-color_r', 'population_n',
      'population_v', 'habitat_p', 'habitat_u', 'habitat_w']

    data = np.array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])
    logging.info('Empty numpy array created')

    if request.method == 'POST':
        cap_color  = request.form['cap-color']
        if cap_color == 'n':
            data[0]= (variable) request: Request

        odor       = request.form['odor']
        if odor == 'c':
            data[1]=1
```



```
<form action="/submit" method = 'post'>
    <label for="cap-color">cap-color:</label><tr>
    <select id="cap-color" name ='cap-color'><tr>
      <option value="n">brown</option>
      <option value="">others</option>
      </select>&ensp;&ensp;&ensp;

    <label for="odor">odor:</label><tr>
    <select id="odor" name ='odor'><tr>
      <option value="c">creosote</option>
      <option value="f">foul</option>
      <option value="l">anise</option>
      <option value="m">musty</option>
      <option value="n">none</option>
      <option value="p">pungent</option>
      <option value="s">spicy</option>
      <option value="y">fishy</option>
    </select>&ensp;&ensp;&ensp;

    <label for="gill-spacing">gill-spacing:</label><tr>
    <select id="gill-spacing" name ='gill-spacing'>
      <option value="w">crowded</option>
      <option value="">others</option>
```

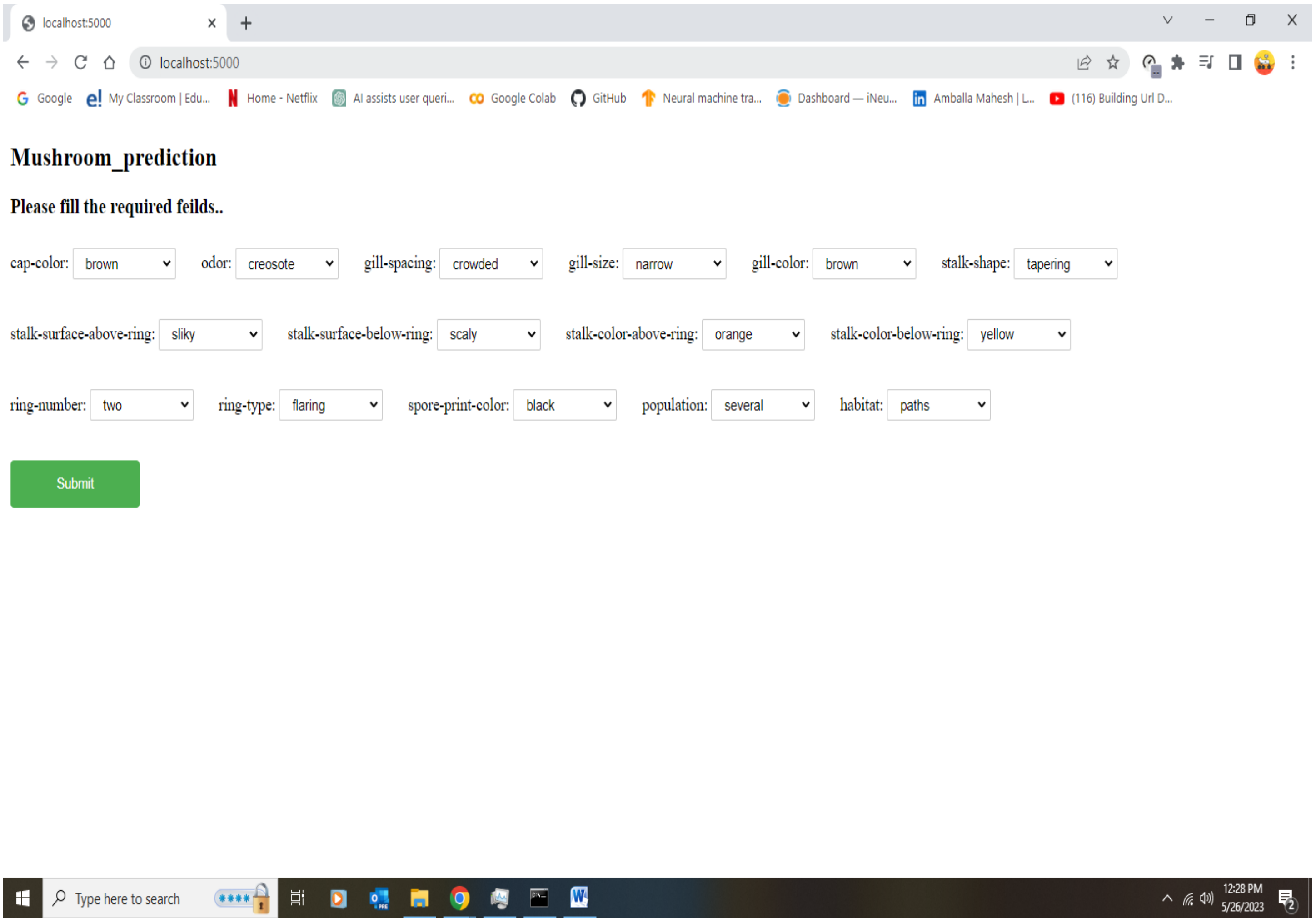


```
<!DOCTYPE html>
<html>
    <h2 style = color:blue;>Final Result...</h2>
<body>

 {% if result == 0 %}
 <h3 style = color:red;>Classification of Mushroom is - Posionous Mushroom!</h3>
</div>
 {% else %}
 <h3 style = color:green;>Classification of Mushroom is - Edible Mushroom!</h3>

 {% endif %}

<a href="http://127.0.0.1:5000/">click here for the test again..</a><br>

</body>
</html>

```

3.6 Testing the Model

- ✓ Verify whether the application is the loading on the local server instance.
- ✓ Verify whether the user can access the application.
- ✓ Verify the user can access the different fields for selection and can be visible
- ✓ Once the user selection the fields and made the submit
- ✓ Check the user can get the result or prediction.
- ✓ Once he gets the prediction.
- ✓ Check the data form the user and prediction from the model is loaded into the local MySQL

- ✓ Check the database and download the data…



localhost:5000

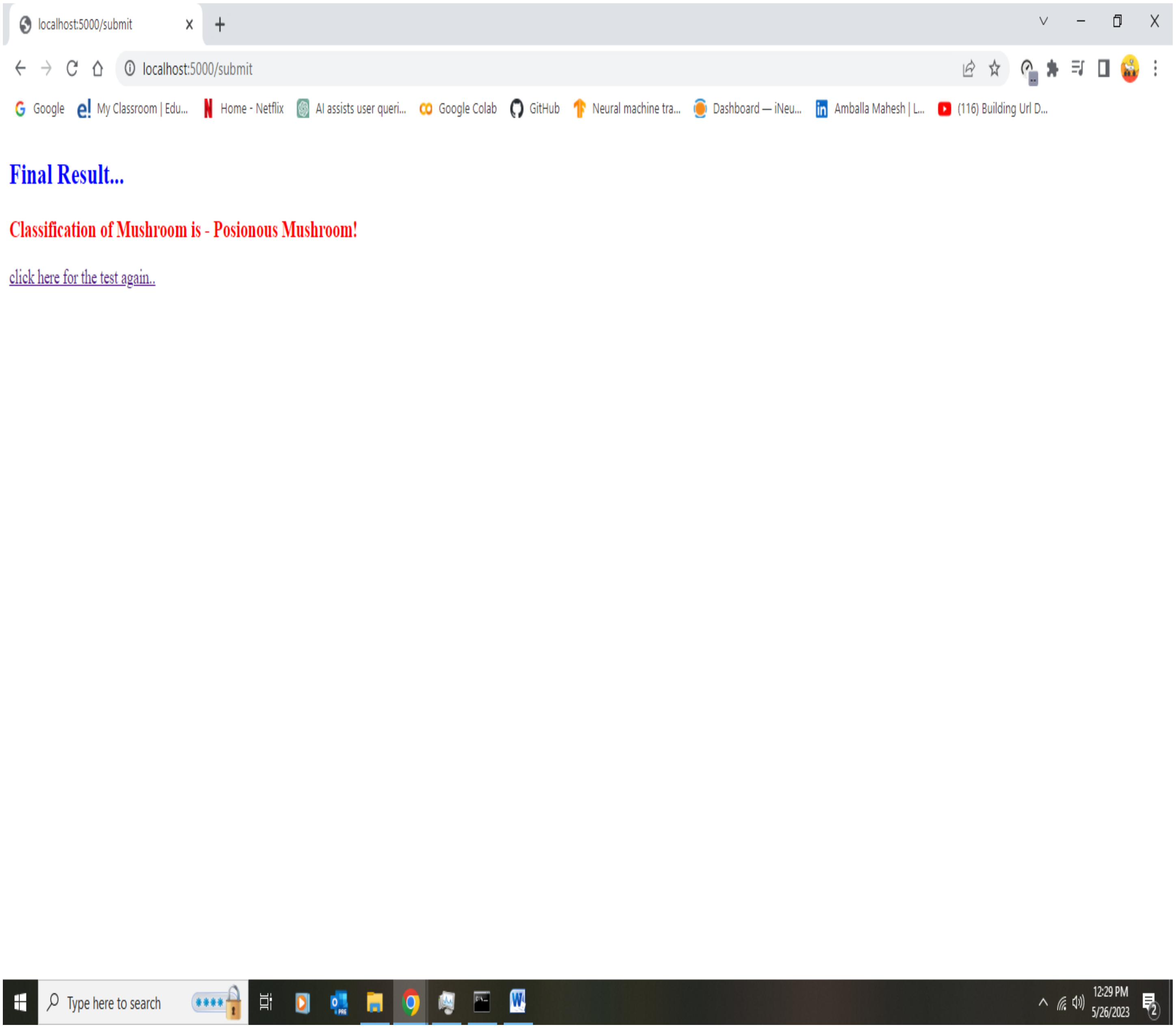
Mushroom_prediction

Please fill the required feilds..

cap-color: brown | odor: creosote | gill-spacing: crowded | gill-size: narrow | gill-color: brown | stalk-shape: tapering

stalk-surface-above-ring: sliky | stalk-surface-below-ring: scaly | stalk-color-above-ring: orange | stalk-color-below-ring: yellow

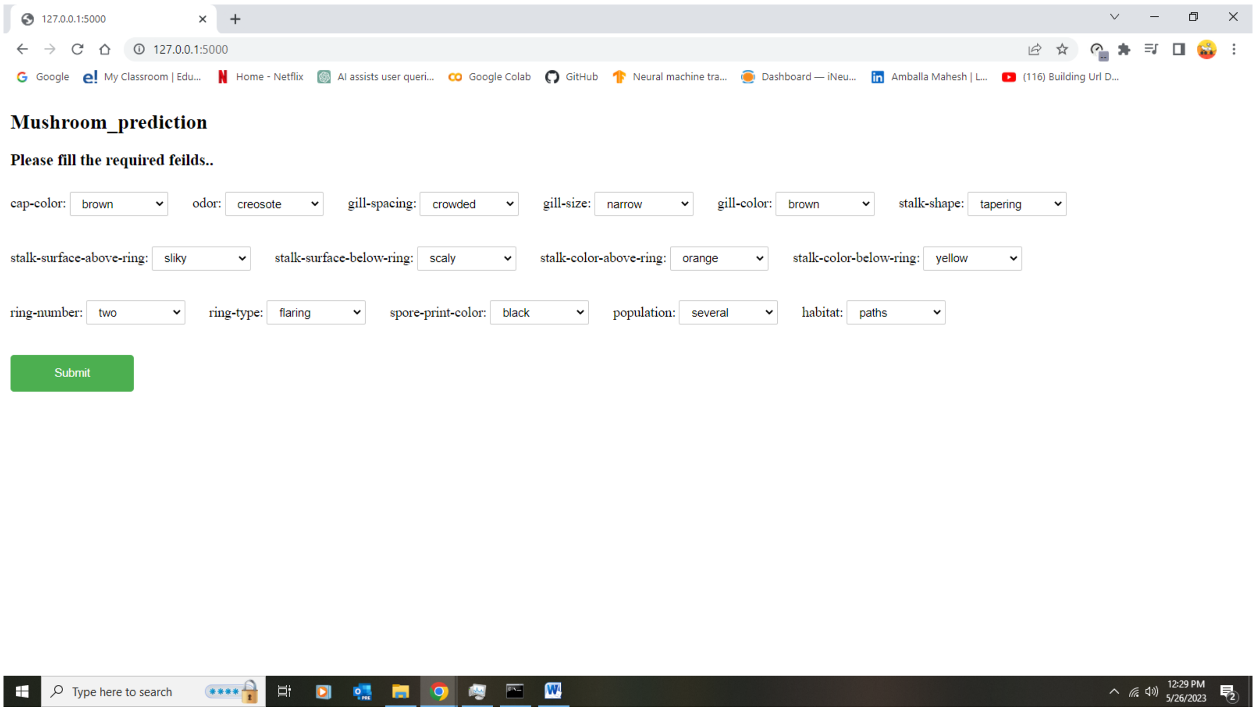
ring-number: two | ring-type: flaring | spore-print-color: black | population: several | habitat: paths


localhost:5000/submit

Final Result...

Classification of Mushroom is - Posionous Mushroom!

click here for the test again..



127.0.0.1:5000

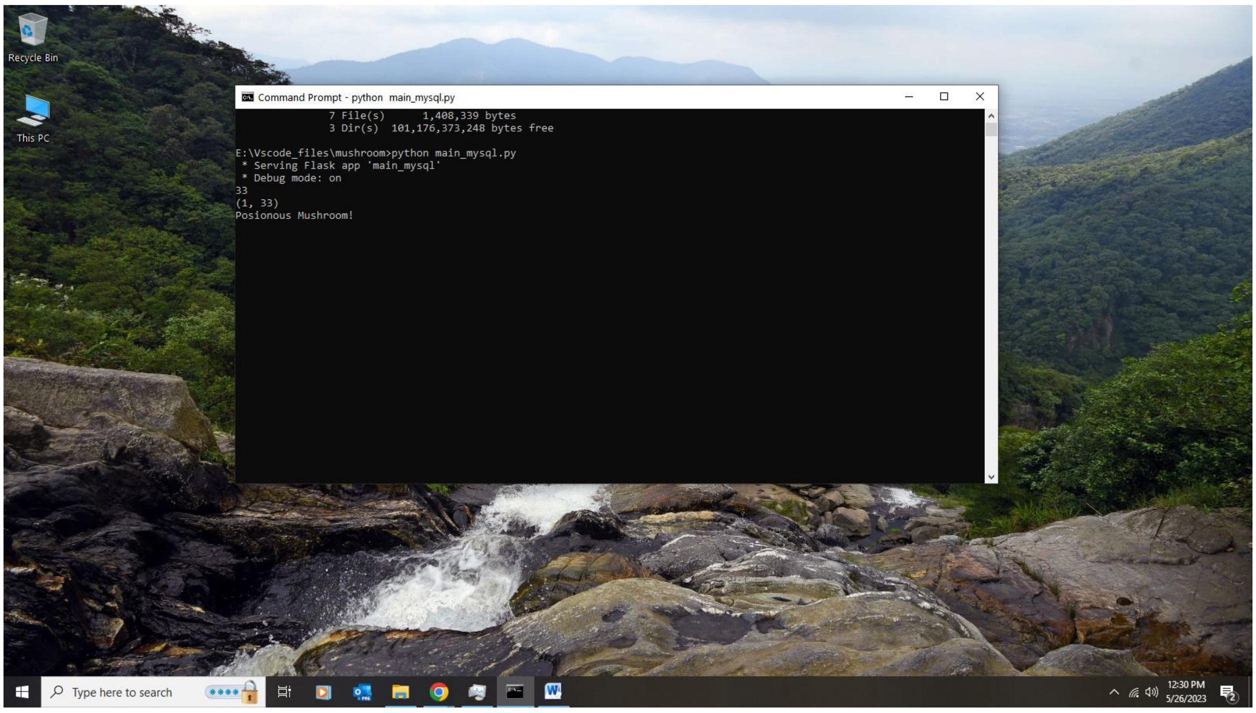
Mushroom_prediction

Please fill the required feilds..

cap-color: brown odor: creosote gill-spacing: crowded gill-size: narrow gill-color: brown stalk-shape: tapering

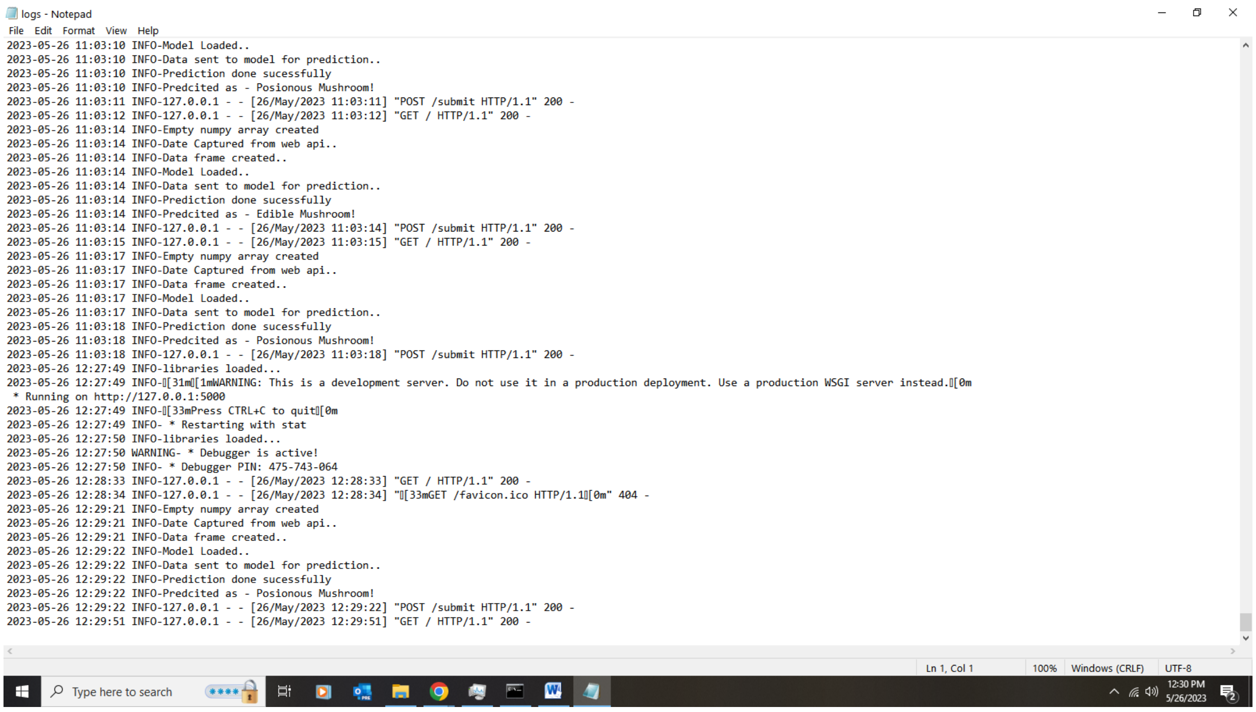
stalk-surface-above-ring: sliky stalk-surface-below-ring: scaly stalk-color-above-ring: orange stalk-color-below-ring: yellow

ring-number: two ring-type: flaring spore-print-color: black population: several habitat: paths

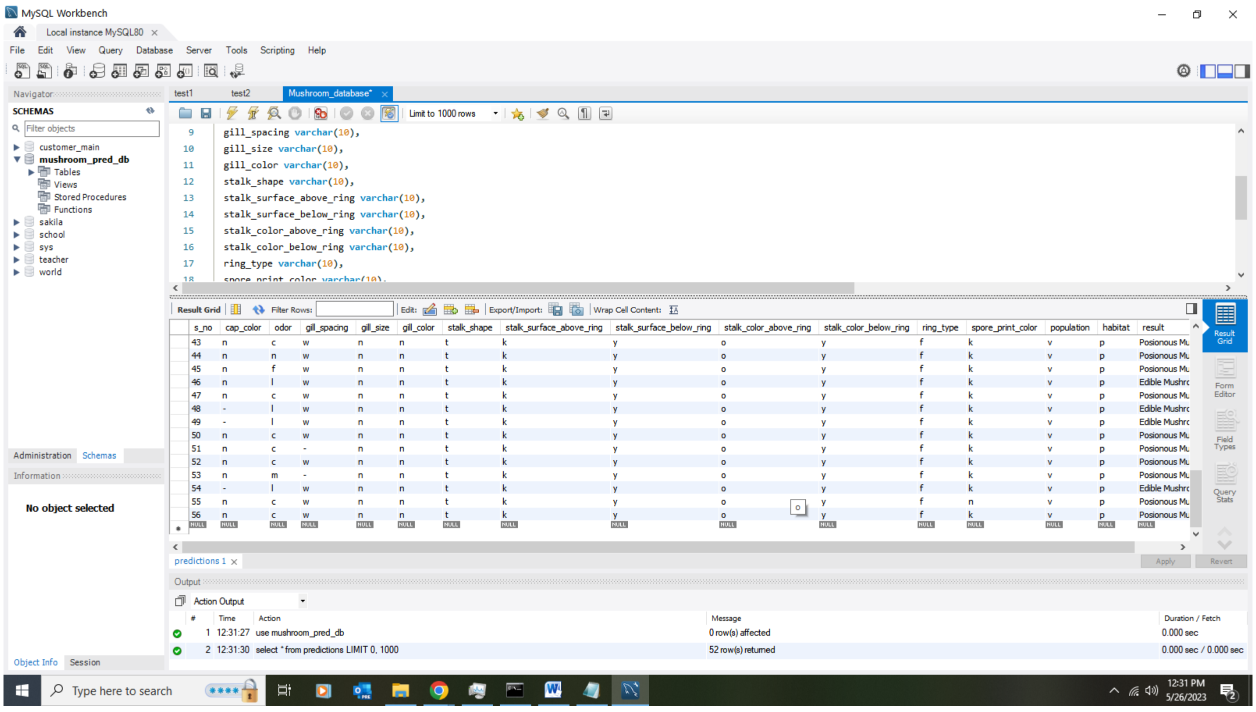


Command Prompt - python main_mysql.py

```
               7 File(s)      1,408,339 bytes
               3 Dir(s)  101,176,373,248 bytes free

E:\Vscode_files\mushroom>python main_mysql.py
 * Serving Flask app 'main_mysql'
 * Debug mode: on
33
(1, 33)
Posionous Mushroom!
```



```
2023-05-26 11:03:10 INFO-Model Loaded..
2023-05-26 11:03:10 INFO-Data sent to model for prediction..
2023-05-26 11:03:10 INFO-Prediction done sucessfully
2023-05-26 11:03:10 INFO-Predcited as - Posionous Mushroom!
2023-05-26 11:03:11 INFO-127.0.0.1 - - [26/May/2023 11:03:11] "POST /submit HTTP/1.1" 200 -
2023-05-26 11:03:12 INFO-127.0.0.1 - - [26/May/2023 11:03:12] "GET / HTTP/1.1" 200 -
2023-05-26 11:03:14 INFO-Empty numpy array created
2023-05-26 11:03:14 INFO-Date Captured from web api..
2023-05-26 11:03:14 INFO-Data frame created..
2023-05-26 11:03:14 INFO-Model Loaded..
2023-05-26 11:03:14 INFO-Data sent to model for prediction..
2023-05-26 11:03:14 INFO-Prediction done sucessfully
2023-05-26 11:03:14 INFO-Predcited as - Edible Mushroom!
2023-05-26 11:03:14 INFO-127.0.0.1 - - [26/May/2023 11:03:14] "POST /submit HTTP/1.1" 200 -
2023-05-26 11:03:15 INFO-127.0.0.1 - - [26/May/2023 11:03:15] "GET / HTTP/1.1" 200 -
2023-05-26 11:03:17 INFO-Empty numpy array created
2023-05-26 11:03:17 INFO-Date Captured from web api..
2023-05-26 11:03:17 INFO-Data frame created..
2023-05-26 11:03:17 INFO-Model Loaded..
2023-05-26 11:03:17 INFO-Data sent to model for prediction..
2023-05-26 11:03:18 INFO-Prediction done sucessfully
2023-05-26 11:03:18 INFO-Predcited as - Posionous Mushroom!
2023-05-26 11:03:18 INFO-127.0.0.1 - - [26/May/2023 11:03:18] "POST /submit HTTP/1.1" 200 -
2023-05-26 12:27:49 INFO-libraries loaded...
2023-05-26 12:27:49 INFO-[31m[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.[0m
 * Running on http://127.0.0.1:5000
2023-05-26 12:27:49 INFO-[33mPress CTRL+C to quit[0m
2023-05-26 12:27:49 INFO- * Restarting with stat
2023-05-26 12:27:50 INFO-libraries loaded...
2023-05-26 12:27:50 WARNING- * Debugger is active!
2023-05-26 12:27:50 INFO- * Debugger PIN: 475-743-064
2023-05-26 12:28:33 INFO-127.0.0.1 - - [26/May/2023 12:28:33] "GET / HTTP/1.1" 200 -
2023-05-26 12:28:34 INFO-127.0.0.1 - - [26/May/2023 12:28:34] "[33mGET /favicon.ico HTTP/1.1[0m" 404 -
2023-05-26 12:29:21 INFO-Empty numpy array created
2023-05-26 12:29:21 INFO-Date Captured from web api..
2023-05-26 12:29:21 INFO-Data frame created..
2023-05-26 12:29:22 INFO-Model Loaded..
2023-05-26 12:29:22 INFO-Data sent to model for prediction..
2023-05-26 12:29:22 INFO-Prediction done sucessfully
2023-05-26 12:29:22 INFO-Predcited as - Posionous Mushroom!
2023-05-26 12:29:22 INFO-127.0.0.1 - - [26/May/2023 12:29:22] "POST /submit HTTP/1.1" 200 -
2023-05-26 12:29:51 INFO-127.0.0.1 - - [26/May/2023 12:29:51] "GET / HTTP/1.1" 200 -
```



```
gill_spacing varchar(10),
gill_size varchar(10),
gill_color varchar(10),
stalk_shape varchar(10),
stalk_surface_above_ring varchar(10),
stalk_surface_below_ring varchar(10),
stalk_color_above_ring varchar(10),
stalk_color_below_ring varchar(10),
ring_type varchar(10),
spore_print_color varchar(10),
```

s_no	cap_color	odor	gill_spacing	gill_size	gill_color	stalk_shape	stalk_surface_above_ring	stalk_surface_below_ring	stalk_color_above_ring	stalk_color_below_ring	ring_type	spore_print_color	population	habitat	result
43	n	c	w	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
44	n	n	w	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
45	n	f	w	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
46	n	l	w	n	n	t	k	y	o	y	f	k	v	p	Edible Mushrc
47	n	c	w	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
48	-	l	w	n	n	t	k	y	o	y	f	k	v	p	Edible Mushrc
49	-	l	w	n	n	t	k	y	o	y	f	k	v	p	Edible Mushrc
50	n	c	w	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
51	n	c	-	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
52	n	c	w	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
53	n	m	-	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
54	-	l	w	n	n	t	k	y	o	y	f	k	v	p	Edible Mushrc
55	n	c	w	n	n	t	k	y	o	y	f	n	v	p	Posionous Mu
56	n	c	w	n	n	t	k	y	o	y	f	k	v	p	Posionous Mu
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Action Output

	#	Time	Action	Message	Duration / Fetch
✓	1	12:31:27	use mushroom_pred_db	0 row(s) affected	0.000 sec
✓	2	12:31:30	select * from predictions LIMIT 0, 1000	52 row(s) returned	0.000 sec / 0.000 sec

4. Key performance indicators (KPI)

- Time and work load reduction by using the flask model.
- Compare the accuracy of model using prediction and actual results.
- Check for the wrong predictions
- If found any wrong predictions again train the model with the new data along with previous data
- Retest the model unless the productions attain the good results.