

**CSC210 Mini Project ll**

**Final Report On Mushroom Classification**

Group members:

Pema Chozom, 12210024

Sonam Choden, 12210030

Dal Bdr Rai, 12210004

Deki Lhazom, 12210048

Tenzin Namgyal Yeshey, 12210099

**Supervisor :** Ms. Tawmo

# Acknowledgement

We would like to extend our sincere gratitude and appreciation to everyone who helped this project report be completed successfully. Their assistance, direction, and encouragement have been crucial in crafting this project.

We would first like to express our sincere gratitude to our project supervisor, Ms. Tawmo, for her invaluable direction and ongoing support during the course of this project. The knowledge, persistence, and sage advice were significant in guiding the project in the appropriate direction. We want to thank her sincerely for the time and work she has put into helping us. We are immensely appreciative of her dedication and willingness to make time for our needs and priorities. Time is a precious resource, and the willingness to give it to us shows how much she cares and supports us. Our growth and development have been greatly impacted by her capacity to prioritize and make room for our worries, inquiries, and debates.

In conclusion, we would like to convey our sincere appreciation for the commitment and for making time for us. We sincerely appreciate her persistent support and acknowledge the huge difference her dedication to our success and welfare has made.

# Final Report On Mushroom Classification

## First Dataset

Firstly, we collected a dataset from Kaggle which was contributed to the UCI Machine Learning repository approximately 30 years ago. That dataset of mushroom characteristics makes us learn from the physical features that are most likely to be edible and which spell certain death. This dataset was drawn from the Audubon Society Field Guide to North American Mushrooms (1981) and contains descriptions of hypothetical samples that correspond to 23 species of gilled mushrooms in the Agaricus and Lepiota Family Mushroom. Each species is classified as either unquestionably edible, unquestionably poisonous, or maybe edible but not advised. The toxic class was joined with the other class.

The dataset contains the following features:

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

1. cap-shape:

bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s

1. cap-surface:

fibrous=f,grooves=g,scaly=y,smooth=s

1. cap-color:

brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y

1. bruises:

bruises=t,no=f

1. odor:

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

1. gill-attachment:

attached=a,descending=d,free=f,notched=n

1. gill-spacing:

close=c,crowded=w,distant=d

1. gill-size:

broad=b,narrow=n

1. 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

1. stalk-shape:

enlarging=e,tapering=t

1. stalk-root:

bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?

1. stalk-surface-above-ring:

fibrous=f,scaly=y,silky=k,smooth=s

1. stalk-surface-below-ring:

fibrous=f,scaly=y,silky=k,smooth=s

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

partial=p,universal=u

1. veil-color:

brown=n,orange=o,white=w,yellow=y

1. ring-number:

none=n,one=o,two=t

1. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
2. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y
3. population:

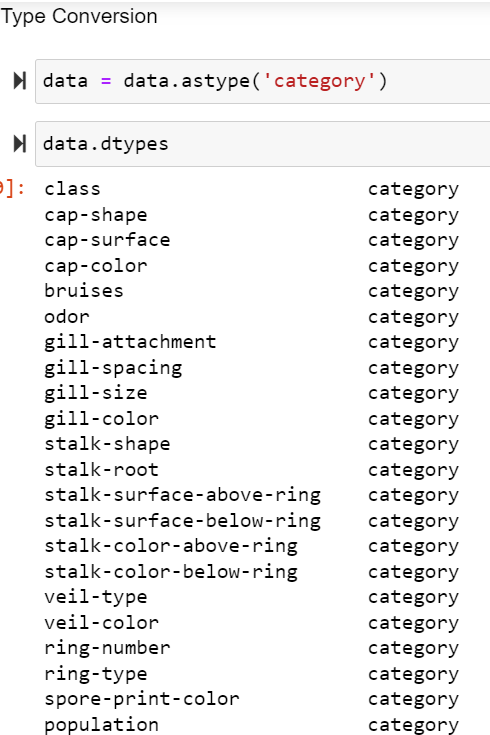
abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y

1. habitat:

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

Originally there were 22 columns excluding class (target feature) and 8124 rows(data) which were of object types. And in this dataset, there were no duplicates and no null values. As each unique value in the object data type is kept as a separate Python object, which adds a lot of costs. In contrast, category types express unique values using integer-based codes, which significantly reduces the amount of memory needed, especially when working with huge datasets. In order to reduce memory space we converted the object datatype to a categorical data type.

Since there is no relationship between veil-type and the target feature we dropped the column.



We split the data into train and test sets keeping “X” as features excluding target(‘class’) and “Y” as target. We kept 20% of the original data as a test set and the rest 80% as the training set which includes x\_train having 6499 rows and 21 columns and y\_tain having 6499 rows, and x\_test having 1625 rows and 21 columns and y\_test having 1625 rows.

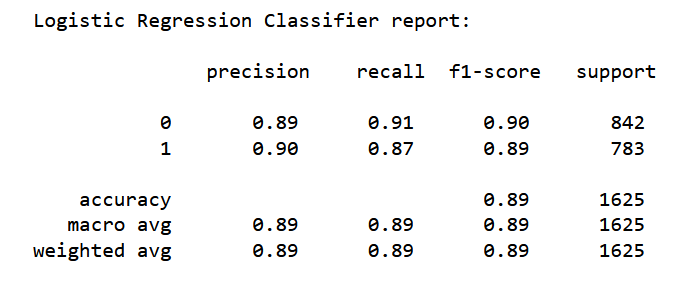
We encoded the training dataset and testing dataset using LabelEncoder as OneHotEnding creates lots of dummy columns making multiple columns in the dataset. We visualized data using a heatmap (by finding a correlation between features).

We used chi\_2 and mutual\_info\_classif as score function of SelectKBest to select features where features are selected based on correlation. By using chi\_2, 10 features were selected namely cap-surface, bruises, gill-spacing, gill-size, gill-color, stalk-root, ring-type, spore-print-color, population, and habitat.

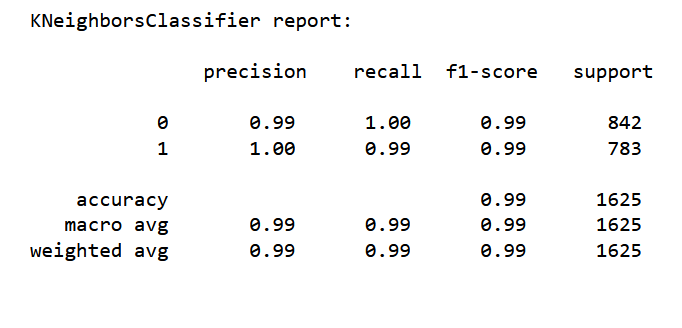
By using mutual\_info\_classif, some different features were selected which include odor, gill-size, gill-color, stalk-surface-above-ring, stalk-surface-below-ring, stalk-color-above-ring, stalk-color-below-ring, ring-type, spore-print-color, and population.

Trained the model using mutual\_info\_classif feature selection. And got the following score using different model:

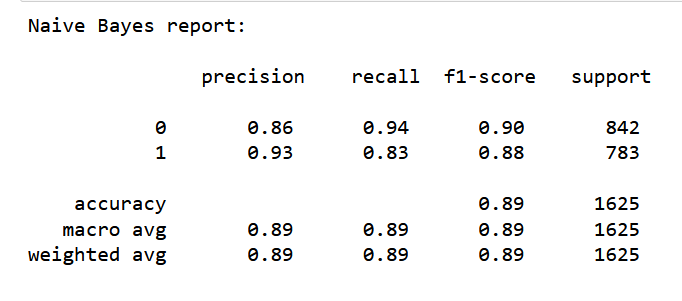
1. Logistic Regression



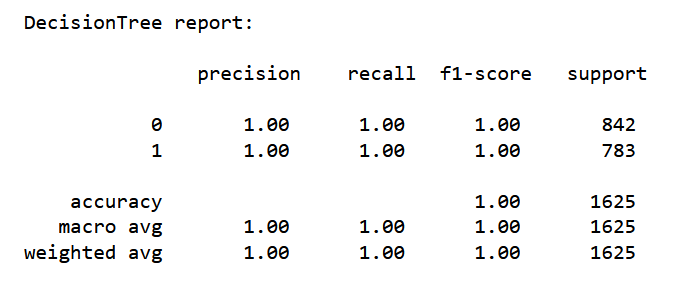
1. KNN Classification



1. Naive Bayes Classification

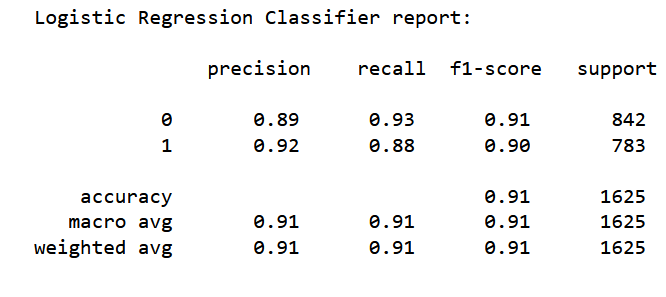


1. Decision Tree

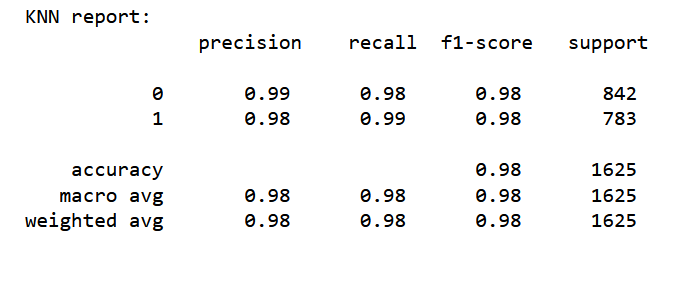


Trained the model using chi\_2 feature selection. And got the following score using different model:

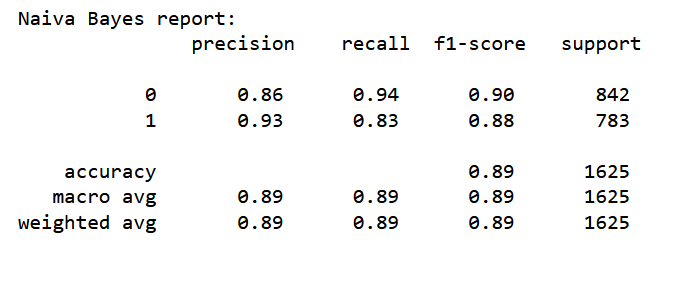
1. Logistic Regression



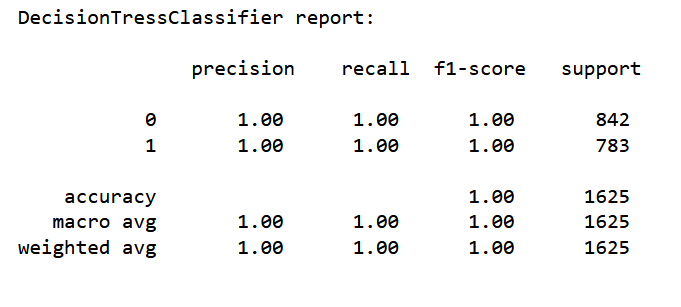
1. KNN Classification



1. Naive Bayes Classification

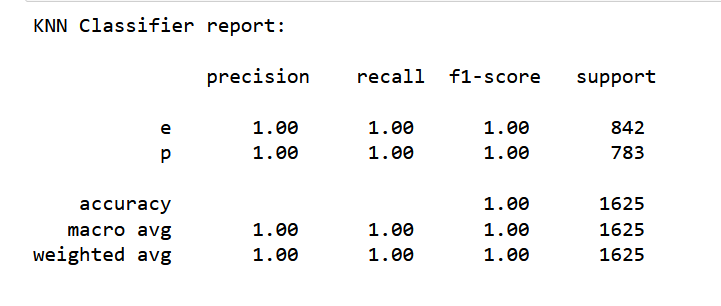


1. Decision Tree

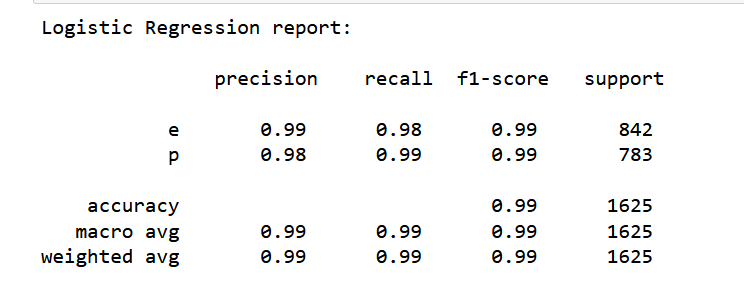


Likewise, we created a pipeline using different models with chi2 feature selection. We created a pipeline using different models mentioned above. The accuracy score for the models are as follows:

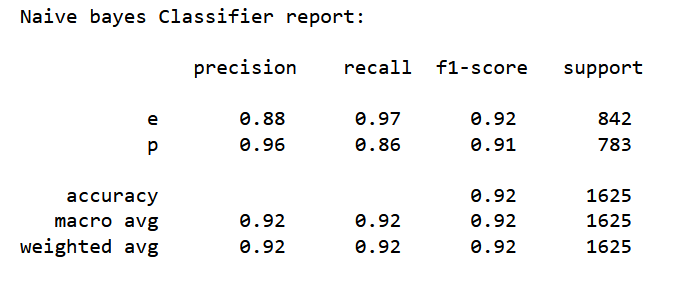
1. KNN



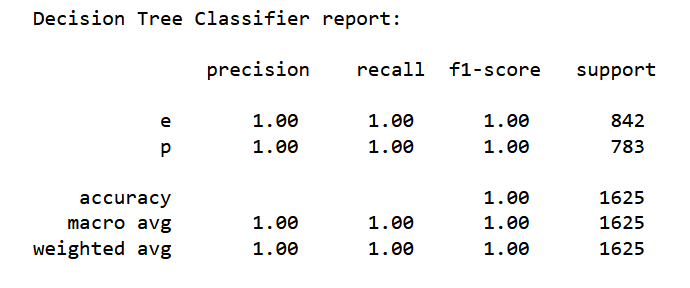
1. Logistic Regression



1. Naive Bayes

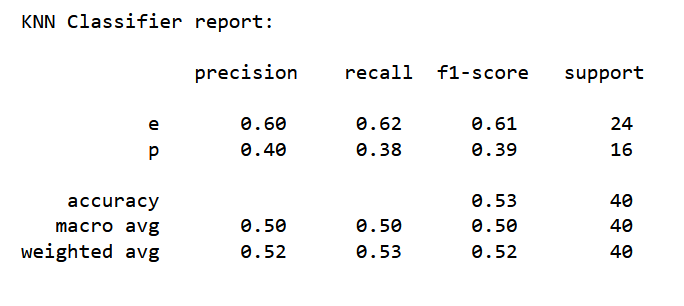


1. Decision Tree

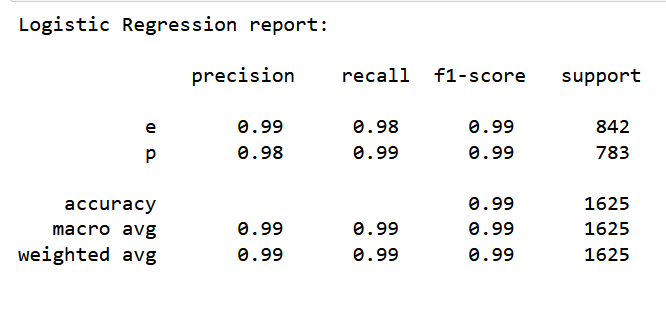


We tried to predict unseen data using the pipeline module and got the accuracy:

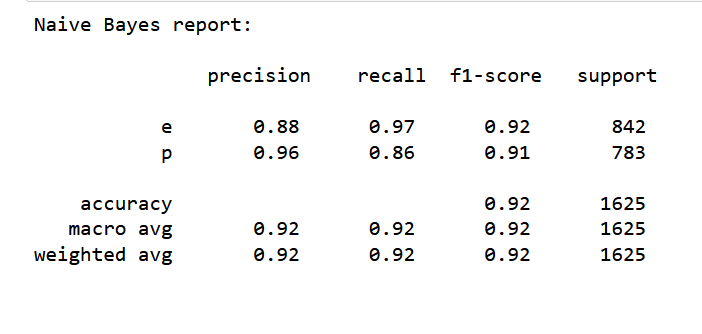
1. KNN



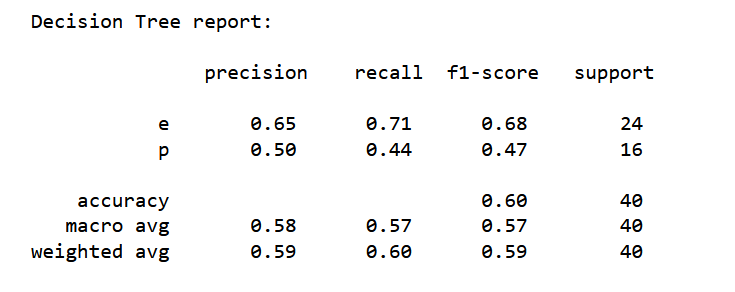
1. Logistic Regression



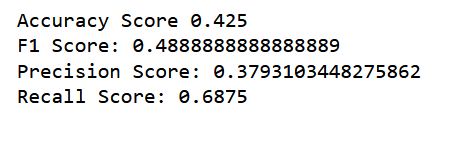
1. Naive Bayes



1. Decision Tree



Since the model was not performing well and it was not able to predict well for unseen data. We did some hyper-tuning and checked the performance for new unseen data. All the models gave the same below score.



## Second Dataset

Secondly, we build models for the new data set as the scores weren’t as expected for the first dataset. The dataset includes species drawn from source book Patrick Hardin.Mushrooms & Toadstools. (Zondervan, 1999).

Based on 173 species and 353 mushrooms per species, the dataset has 61069 potential mushrooms with caps. Each mushroom is classified as either definitely poisonous, definitely edible, or maybe edible but not advised (the latter class was mixed with the poisonous class).

The dataset contains following attributes :

Target feature is a binary class that is divided into edible(e) and poisonous (p). There are 20 features where 3 are metrical (Numerical), while 17 are nominal(Categorical).

1. cap-diameter:

float number in cm

1. cap-shape :

bell=b, conical=c, convex=x, flat=f,sunken=s, spherical=p, others=o

1. cap-surface :

fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, wrinkled=w, fleshy=e

1. cap-color :

brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k

1. does-bruise-bleed :

bruises-or-bleeding=t,no=f

1. gill-attachment :

adnate=a, adnexed=x, decurrent=d, free=e, sinuate=s, pores=p, none=f, unknown=?

1. gill-spacing :

close=c, distant=d, none=f

1. gill-color :

brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k, none=f

1. stem-height :

float number in cm

1. stem-width :

float number in mm

1. stem-root :

bulbous=b, swollen=s, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r

1. stem-surface :

fibrous=i, grooves=g, scaly=y, smooth=s, shiny=h, leathery=l, silky=k, sticky=t, wrinkled=w, fleshy=e, none=f

1. stem-color :

brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k, none=f

1. veil-type :

partial=p, universal=u

1. veil-color :

brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k, none=f

1. has-ring :

ring=t, none=f

1. ring-type :

cobwebby=c, evanescent=e, flaring=r, grooved=g, large=l, pendant=p, sheathing=s, zone=z, scaly=y, movable=m, none=f, unknown=?

1. spore-print-color :

brown=n, buff=b, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, blue=l, orange=o, black=k

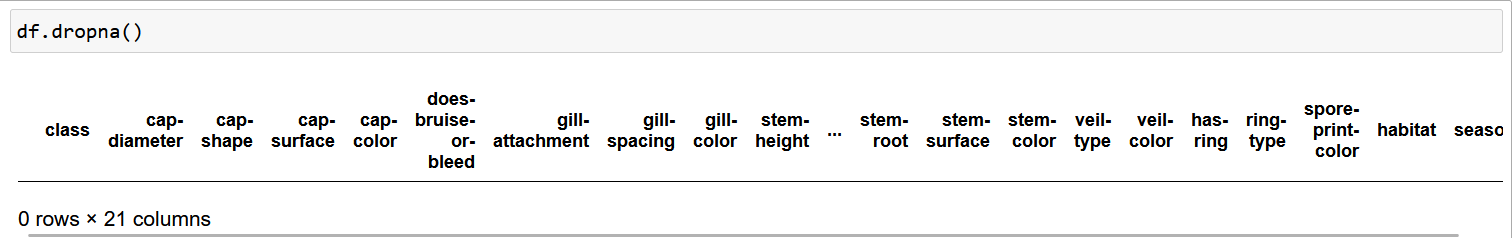
1. habitat :

grasses=g, leaves=l, meadows=m, paths=p, heaths=h, urban=u, waste=w, woods=d

1. season:

spring=s, summer=u, autumn=a, winter=w

We were suggested to drop all the null values but when we drop the null values we get an empty dataframe.



## Combined Dataset:

Since there is no data after dropping the null values. We combined the first dataset and second dataset. We dropped the column having null value for the second dataset. There were only six columns left including the target feature and renamed ‘does-bruise-or-bleed’ to ‘bruises’ to make the column name the same as the first dataset. We concatenated the first dataset and second dataset (having the same column name). The Combined dataset has 69193 rows and 6 columns including target.

The combined dataset contains following features after dropping null values:

1. cap-shape:

bell=b, conical=c, convex=x, flat=f, knobbed=k,sunken=s, spherical=p, others=o

1. cap-color:

brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y, green=r, blue=l, orange=o, black=k

1. bruise

bruises=t, no=f

1. gill-color

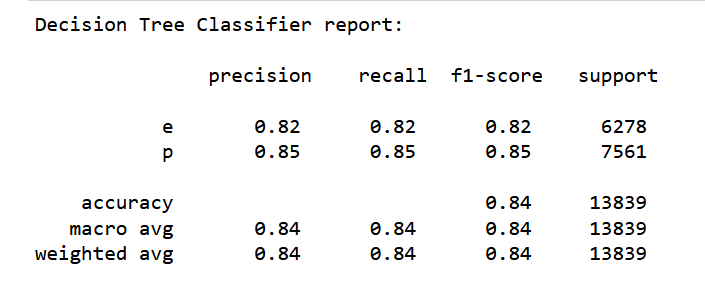
none=f, black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y, blue=l

1. habitat

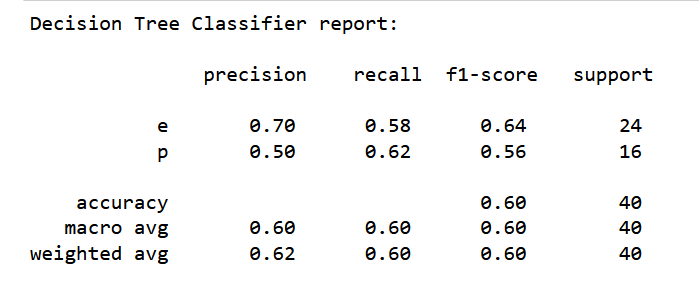
grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d , heath = h

We split the data into train and test sets keeping “x” as features excluding target(‘class’) and “y” as target. We kept 20% of the original data as a test set and the rest 80% as the training set. Then we created the pipeline model. The sequential chain contains imputation with most frequently occurring data and OneHotEncoder for encoding data. The chain also includes Decision Tree as estimator.

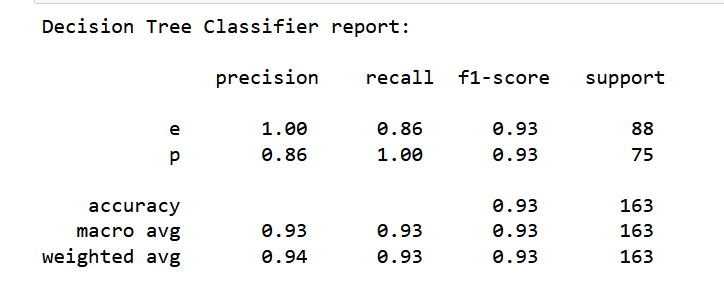
Below shows Decision Tree score:



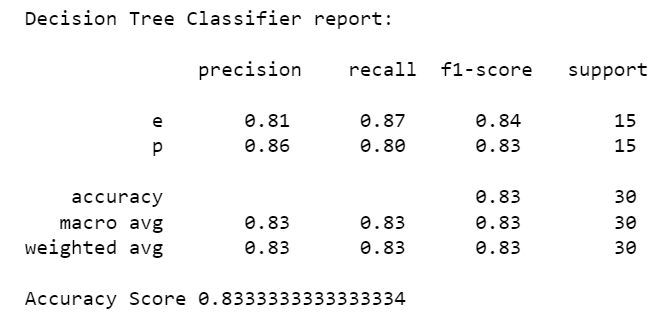
Score for new unseen dataset:



Since the score was little low we tried with data from the second dataset which were not included in testing and training the model and considered it as new unseen data. Below shows score:



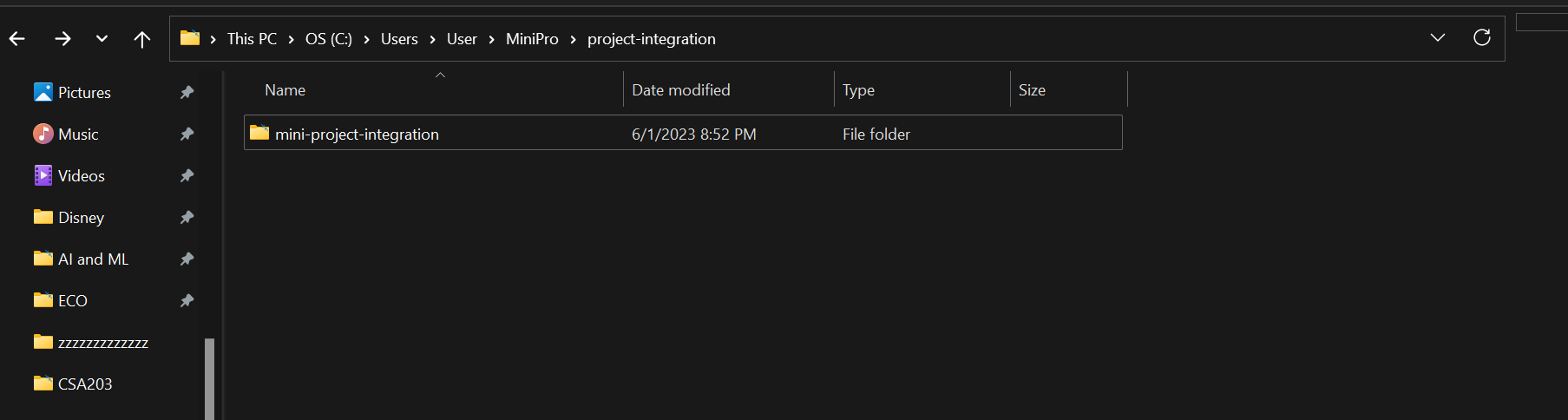
And we also checked score on new collected dataset :



The performance for unseen collected data using combined dataset achieved better performance compared to the first dataset.

# Model integration

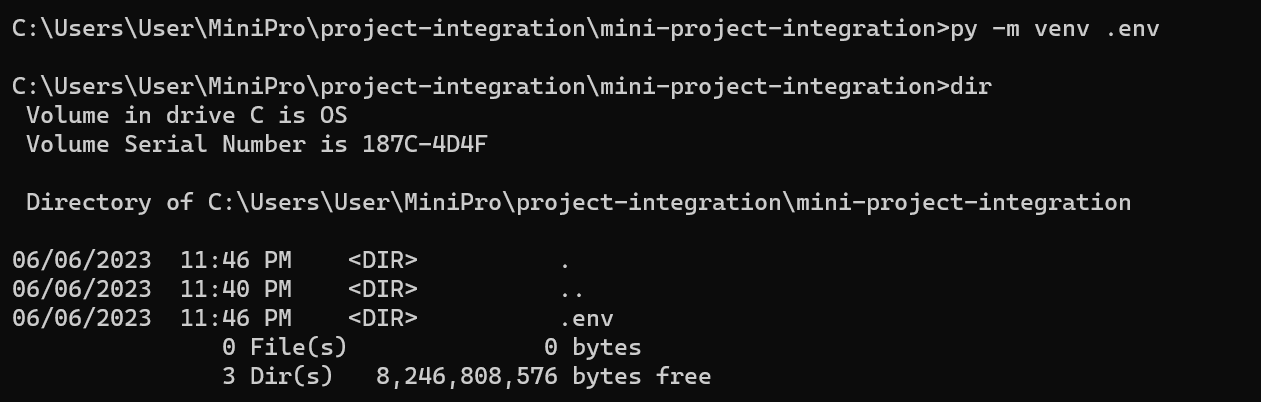
After the pipeline model we downloaded the pickel model fileFirst, a folder relating to the project was created in a location with a name 'mini-project-integration'.



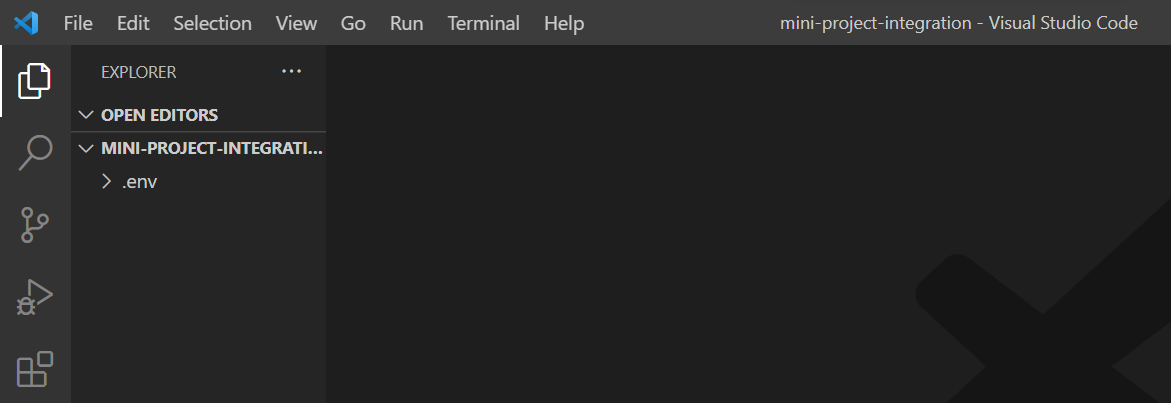
Next, we set up a virtual environment to install the project's necessary packages. Opened CMD and changed the current directory to the newly formed directory to accomplish this. In that directory, we constructed a virtual environment.

To create virtual environment, type following command on the terminal

***py –m venv .virtual environment name***

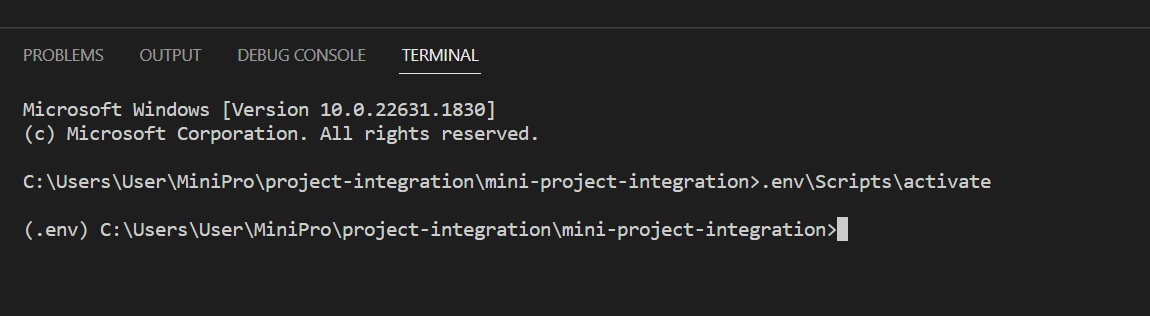


We activated the virtual environment by activating vscode before using it. First, opened our project in virtual environment in vscode.

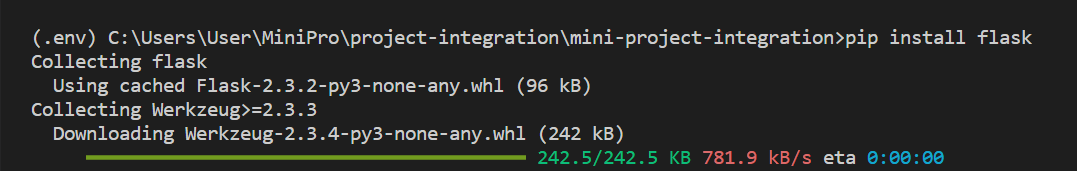


Opened terminal in vscode. Chose command prompt terminal. Activated our virtual environment by running activate file using following command:

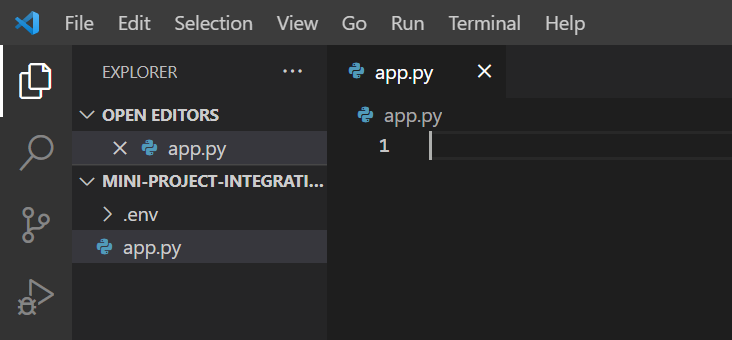
.***env\Scripts\activate***

******

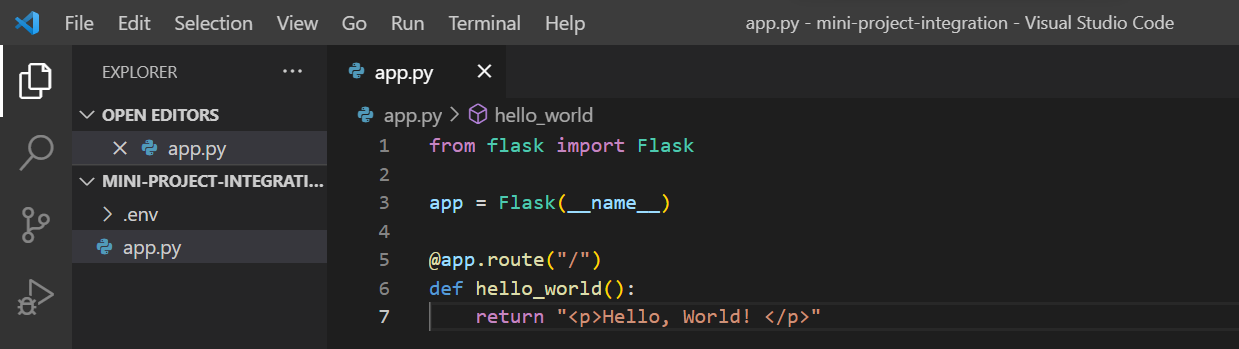
We installed necessary packages (flask) in the virtual environment.



In our project directory, outside the virtual environment, we created a python file. We have created as app.py as shown below.

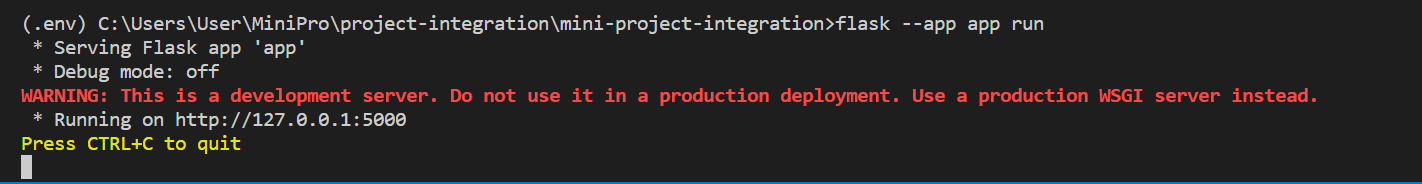


Wrote following code in app.py file to run our first flask app.

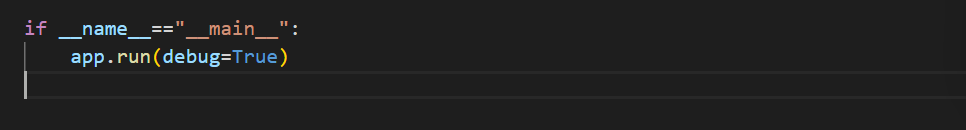


Tested our first flask app. Ran the following command in the terminal

***flask --app filename(app in our case) run***



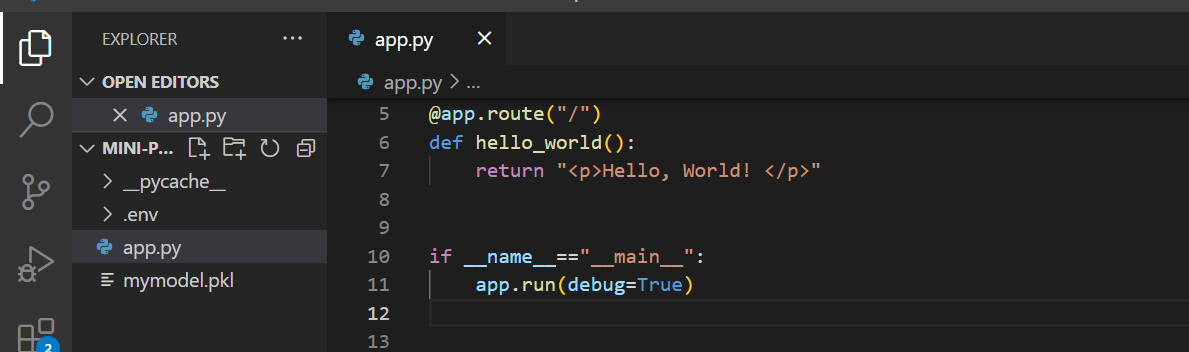
To avoid having to restart our app each time we made changes to our code by adding the following code at the end of our file.



Checked the output in browser

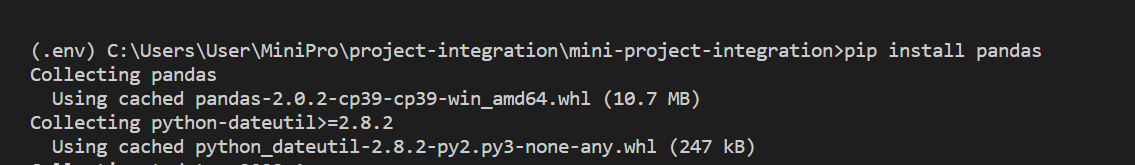


We imported our machine learning model into the Flask application. Saved models should be kept in the project directory .

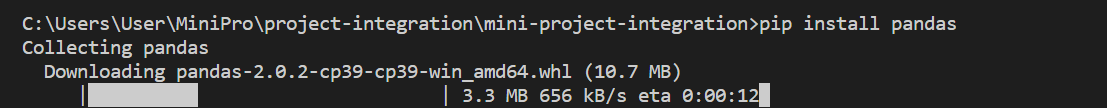


Installed some additional libraries such as numpy, pandas and sklearn.

Installed numpy:

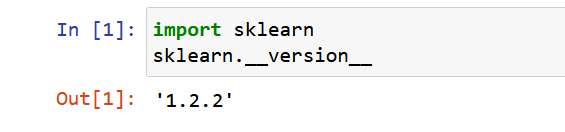


Installed pandas:

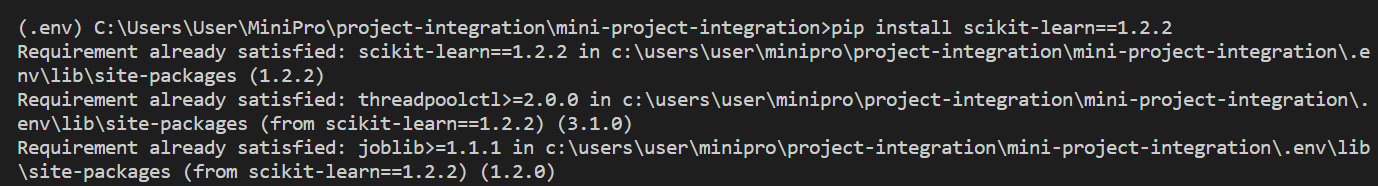


Installed sklearn:

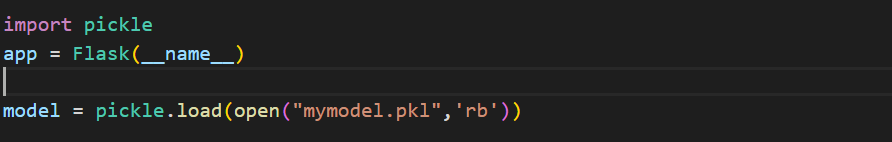
Before installing SkLearn for our project, we made sure to check the version in our Jupyter notebook.



Since, the sklearn’s version is 1.2.2 we installed 1.2.2 version of sklearn in our virtual environment.



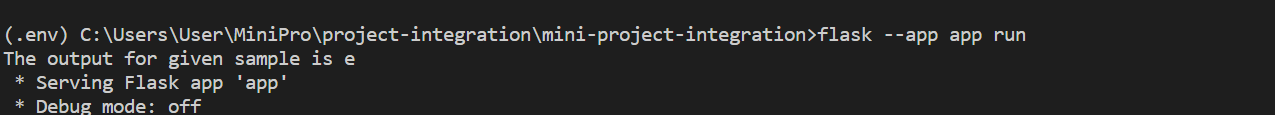
After installing all the required libraries, we loaded our model in our application using pickle module.



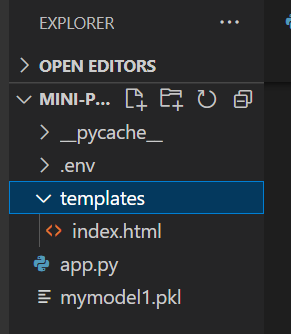
Defined some sample feature and use predict method to test our model in flask app.



Saved the program, we got output in terminal as shown below:



Our model was working as expected. However, do not want to print output in terminal as done in above step. So, we created UI to get our data from user. To do that, we created a templates folder inside project folder. Inside the templates folder create index.html page.



The index.html page contains form fields to get input for all the features from user (also landing page in our case).

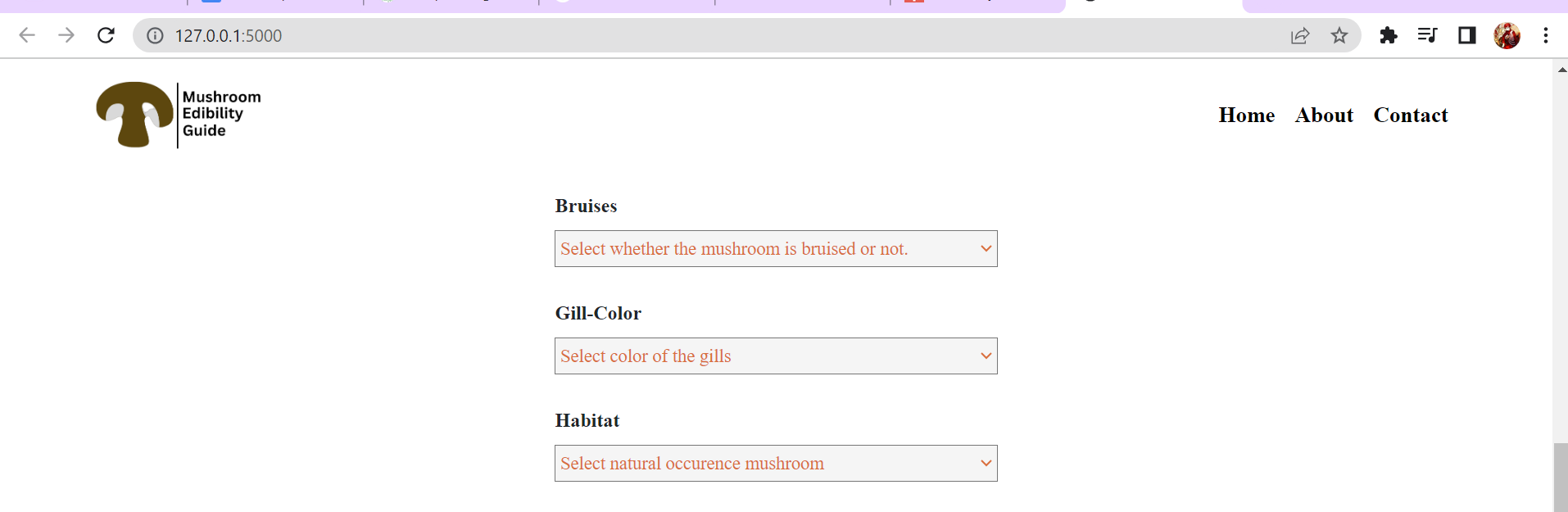
After that, in app.py,we added our index.html page in the route. To do that we import render\_template function from flask. And then changed the route.

| from flask import Flask,render\_template |
| --- |

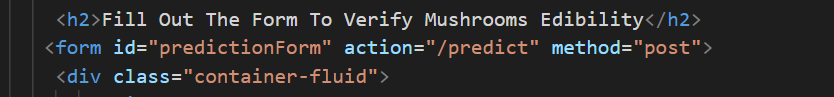
Route to index page

| @app.route("/")  def index():  return render\_template("index.html") |
| --- |

Saved the file and refreshed our browser.



In the browser, our page appeared as expected. To get inputs from form fields, it is required to specify action and method properties of the form element. We specified in the index.html page for this project.



To define new route ‘/predict’ in app.py as defined in action properties of the form element. Since we are getting data from user, we erased the x\_sample created and created route.

| @app.route("/predict",methods=['POST'])  def prediction(): |
| --- |

Inside the prediction() function, we need to get input from user. For this task, we imported request from flask.

| from flask import Flask, render\_template, request  import json  import pickle  import numpy as np  import pandas as pd |
| --- |

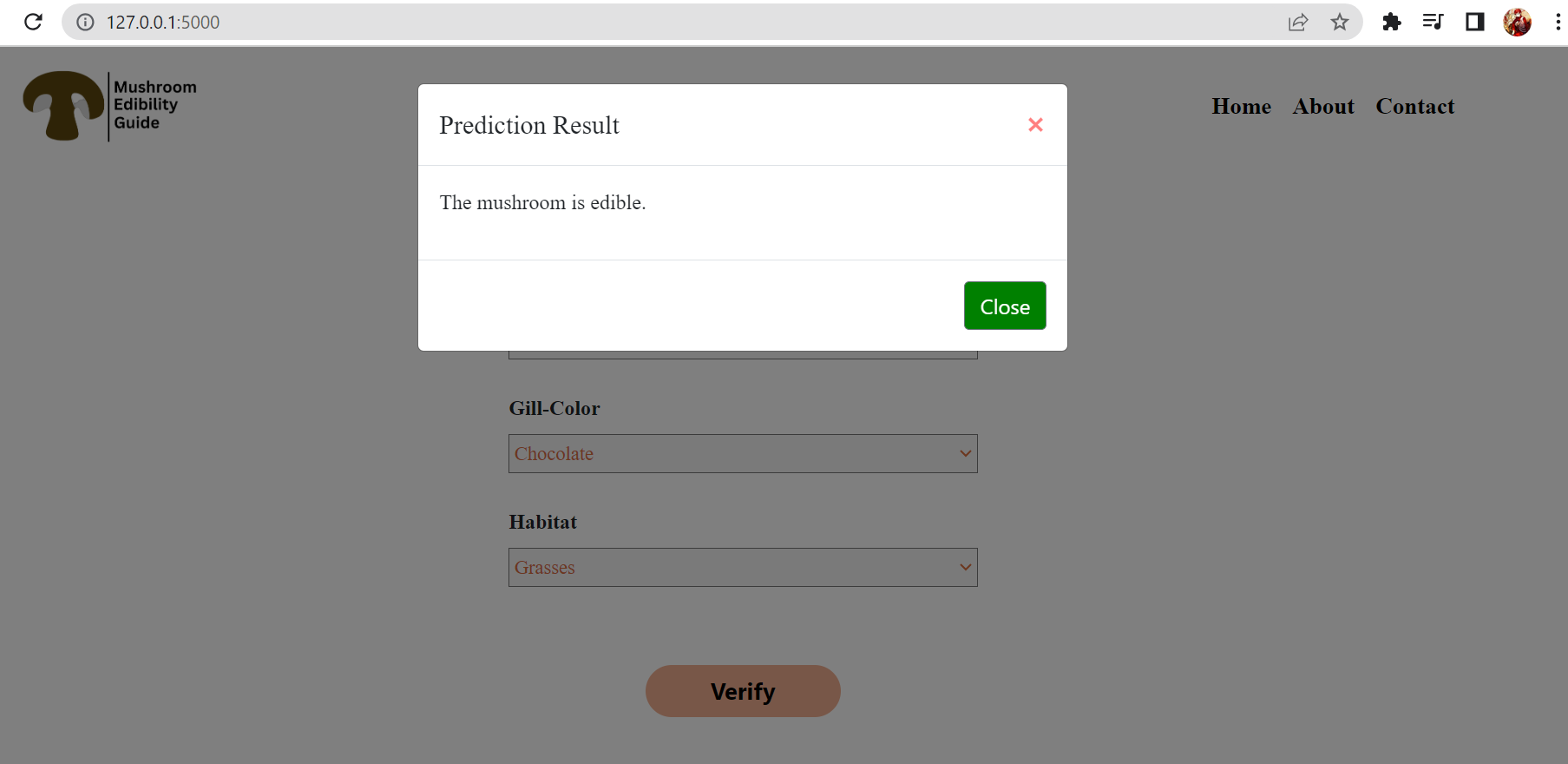
We used request to get data from form inputs from user. The code is shown below.

| @app.route("/predict", methods=['POST'])  def prediction():  if request.method == 'POST':  cap\_shape = request.form['cap\_shape']  cap\_color = request.form['cap\_color']  bruises = request.form['bruises']  gill\_color = request.form['gill\_color']  habitat = request.form['habitat']  sample = [[cap\_shape, cap\_color, bruises, gill\_color, habitat]]  x = pd.DataFrame(sample, columns=cols)  # print(x)  result = model.predict(x)  return render\_template("output.html", value=result) |
| --- |

In the code shown above, we created pandas dataframe from retrieved data. Then used our model to make prediction on data retrieved from user. Then the prediction output is stored in variable result and it is passed to output.html page as variable value. Created output.html with following content. To get result in output.html, jinja is used. To get more useful result, we edited the jinja code using if else condition.

| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Document</title>  </head>  <body>  {% if value[0] == "e"%}  <p>Mushroom has been verified as edible.</p>  {% elif value[0] == "p"%}  <p>Mushroom has been verified as poisonous.</p>  {% endif %}    </body> |
| --- |

We again refreshed browser and entered all the fields of the form and clicked on predict button. When we click on predict button, redirect to “/predict” route and get result 'edible' or 'poisonous'.



# **Conclusion:**

For our project, our final dataset includes 6 features including target. The features consist of cap-shape, cap-surface, bruises, gill-color, habitat and class as target variable. We set out to evaluate and get insights from a significant amount of data in this dataset study (where dataset contains 61193 data). We used advanced machine learning techniques, exploratory data analysis, and data cleansing in our analysis. Among four estimator we chose Decision Tree as our final estimator and the accuracy score of 84% was achieved. To integrate our model with front-end we created pickel file and used flaskenv, where all necessary packages were installed and used accordingly.

# Additional References:

The link below shows our Jupyter file for the first mentioned description above:

<https://colab.research.google.com/drive/1rzfETP9ZnQPpCfjV8gTaYXyp1BCTnKI5?usp=sharing>

The link below shows the Jupyter file for our pipeline model:

<https://colab.research.google.com/drive/11n5DyjxDb26QTsb_-5TEt0jv9BP5xpED?usp=sharing>

We tried to predict unseen data using the pipeline module:

<https://colab.research.google.com/drive/1vP5AxNa0pt_PpYTgISScgllWDXGqjd6u?usp=sharing>

Hyperparameter Tuning:

<https://colab.research.google.com/drive/1_g1LokiSDPB8REs7em7PZdICIEtiQpt6?usp=sharing>

New Dataset after dropping null values:

<https://colab.research.google.com/drive/1YJOGe0x-mJ606NLMXq0Jyi9S4e-u4ZcV>

Model for combined dataset:

<https://colab.research.google.com/drive/1HQ4EYAoqvK1CwR-Lhzxjag6WbStAzFJ7#scrollTo=f67cc487>

Final Trained Model:

<https://colab.research.google.com/drive/1cs9W06VVrUlW7p5OjLIe46O0xJzsheqh#scrollTo=deea3c07>