Dave's Attempt at Mushroom Classification

AWS Machine Learning Specialty Course 7.92

[Pasted -v-]

Exercise - Mushroom Classification

In this exercise, you need to classify mushroom as edible or poisonous.

This data set is provided by UCI: https://archive.ics.uci.edu/ml/datasets/mushroom. You can read the problem description and objective in the UCI website.

Build a classifier using XGBoost. You also need to perform data cleanup and transformation before you can train on XGBoost.

Complete Solution is available here (however, try to solve on your own):

https://github.com/ChandraLingam/AmazonSageMakerCourse/tree/master/xgboost/MushroomClassification

Data Prep and Training in the same Notebook

Follow iris (and, for Q&R, the solution)

Mushroom Classification Dataset

Input features, ready for a Python list:
'cap-shape', 'cap-surface', 'cap-color', 'bruises','odor', 'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color','stalk-shape', 'stalk-root', 'stalk-surface-above-ring','stalk-surface-below-ring', 'stalk-color-above-ring','stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-print-color', 'population', 'habitat'

Target:

Is the mushroom edible? True is edible, False is poisonous

'mushroom_is_edible'

In [1]:
```
!pip install xgboost
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Collecting xgboost
  Downloading xgboost-1.7.6-py3-none-manylinux2014_x86_64.whl (200.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.3/200.3 MB 3.1 MB/s eta 0:00:0000:0100:01
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgb
oost) (1.22.3)
Requirement already satisfied: scipy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgb
oost) (1.10.1)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.6
```

In [2]:
```
!pip install requests
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: requests in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (2.29.
0)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site
-packages (from requests) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (f
rom requests) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-pa
ckages (from requests) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packa
ges (from requests) (2023.5.7)
```

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import os
import sys
import pathlib
import shutil
import itertools
```

```
import requests       # Might need  `pip install requests`
import zipfile as zf

import xgboost as xgb
from sklearn import preprocessing
from sklearn.metrics import classification_report, confusion_matrix
```

In [4]:
```
# For getting the data
data_url = "https://archive.ics.uci.edu/static/public/73/mushroom.zip"
data_zip_filename = "mushroom.zip"
data_filename = "mushroom_all.csv"

# For zipfile
working_dir = os.getcwd()
unzipped_dir = "mushroom_unzipped"
new_dirname = os.path.join(working_dir, unzipped_dir)
pathlib.Path(new_dirname).mkdir(parents=True, exist_ok=True)
```

Checked: The new directory is there. Hooray!

In [5]:
```
print(f"  new_dirname:\n{new_dirname}")
```

```
  new_dirname:
/home/ec2-user/SageMaker/AmazonSageMakerCourse/xgboost/dwb_Mushroom_Try_2023-07-29/mushroom_unzipped
```

In [6]:
```
#  Get the zip - path and fact it's a zip from looking at
#+ https://archive.ics.uci.edu/ml/datasets/mushroom
mushroom_request = requests.get(data_url,
                                allow_redirects=True)
with open(data_zip_filename, 'wb') as fh:
    fh.write(mushroom_request.content)
##endof:  with open ... fh
```

Checked: the new zip archive is there. Hooray again!

Not sure about this next part, but I'm following Chandra's stuff as well as I know how.

In [7]:
```
zipfile_thing = zf.ZipFile("mushroom.zip")
zipfile_thing.extractall(new_dirname)
```

Checked: the contents of the zip are there. ~~I've learned that `zipfile_thing.extractall(".")` isn't the way to go~~ - that or I did something wrong on my local machine. Yeah, oops, I just tried it here on AWS, and it worked fine. It's nice to have it in another directory, though, especially since I want to change the name.

In [8]:
```
# bash is underneath this Notebook
!ls -lah "/home/ec2-user/SageMaker/AmazonSageMakerCourse/xgboost/dwb_Mushroom_Try_2023-07-29/mushroom_unzipped"
```

```
total 508K
drwxrwxr-x 3 ec2-user ec2-user 4.0K Jul 30 14:52 .
drwxrwxr-x 4 ec2-user ec2-user 4.0K Jul 31 02:59 ..
-rw-rw-r-- 1 ec2-user ec2-user 365K Jul 31 02:59 agaricus-lepiota.data
-rw-rw-r-- 1 ec2-user ec2-user 6.7K Jul 31 02:59 agaricus-lepiota.names
-rw-rw-r-- 1 ec2-user ec2-user 109K Jul 31 02:59 expanded.Z
-rw-rw-r-- 1 ec2-user ec2-user  193 Jul 31 02:59 Index
drwxrwxr-x 2 ec2-user ec2-user 4.0K Jul 30 14:52 .ipynb_checkpoints
-rw-rw-r-- 1 ec2-user ec2-user  853 Jul 31 02:59 README
```

In [9]:
```
!stat "/home/ec2-user/SageMaker/AmazonSageMakerCourse/xgboost/dwb_Mushroom_Try_2023-07-29/mushroom_unzipped/expanded.
print()
print("I've done some more inspection on  expanded.Z ;")
print("it's an archive, but not a zip.")
print("Inside is a text file with longer names for characteristics.")
print("I'll skip it for now.")
```

```
  File: ‘/home/ec2-user/SageMaker/AmazonSageMakerCourse/xgboost/dwb_Mushroom_Try_2023-07-29/mushroom_unzipped/expand
ed.Z’
  Size: 111577          Blocks: 224        IO Block: 4096   regular file
Device: 10303h/66307d   Inode: 131485      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/ec2-user)   Gid: ( 1000/ec2-user)
Access: 2023-07-30 14:45:41.315698138 +0000
Modify: 2023-07-31 02:59:52.286338719 +0000
Change: 2023-07-31 02:59:52.286338719 +0000
 Birth: -

I've done some more inspection on  expanded.Z ;
it's an archive, but not a zip.
Inside is a text file with longer names for characteristics.
I'll skip it for now.
```

In [10]:
```
# Check for the file we want.
candidate_fname_1 = os.path.join(new_dirname,
```

```
                                  "agaricus-lepiota.data")
candidate_fname_2 = os.path.join(new_dirname,
                                  "agaricus-lepiota.names")


def head_filename(this_fname, n_lines_to_read=10):
    print("-" * 50)
    print(f"  First {n_lines_to_read} lines from" +
          f"\n{this_fname}")
    print("-" * 5)
    with open(this_fname, 'r', encoding="utf-8") as fh1:
        for i in range(n_lines_to_read):
            # I won't use the i, but Q&R, whatever
            print(fh1.readline())
        ##endof:  for i in range(n_lines_to_read)
    ##endof:  with open ... fh1
    print("-" * 50)
    print()
##endof:  head_filename(<params>)

print()
head_filename(candidate_fname_1)
print()
head_filename(candidate_fname_2)
```

```
--------------------------------------------------
  First 10 lines from
/home/ec2-user/SageMaker/AmazonSageMakerCourse/xgboost/dwb_Mushroom_Try_2023-07-29/mushroom_unzipped/agaricus-lepiot
a.data
-----
p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u

e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g

e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m

p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u

e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g

e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g

e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m

e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m

p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g

e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m

--------------------------------------------------


--------------------------------------------------
  First 10 lines from
/home/ec2-user/SageMaker/AmazonSageMakerCourse/xgboost/dwb_Mushroom_Try_2023-07-29/mushroom_unzipped/agaricus-lepiot
a.names
-----
1. Title: Mushroom Database



2. Sources:

    (a) Mushroom records drawn from The Audubon Society Field Guide to North

        American Mushrooms (1981). G. H. Lincoff (Pres.), New York: Alfred
```

```
        A. Knopf

    (b) Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)

    (c) Date: 27 April 1987


3. Past Usage:

---------------------------------------------------
```

We want `agaricus-lepiota.data`. That's not a huge surprise.

In [11]:
```
shutil.copy(candidate_fname_1, data_filename)
```

Out[11]:
```
'mushroom_all.csv'
```

In [12]:
```
# checking; might as well use that head_filename function
head_filename(data_filename)
```

```
--------------------------------------------------
  First 10 lines from
mushroom_all.csv
-----
p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u

e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g

e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m

p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u

e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g

e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g

e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m

e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m

p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g

e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m

--------------------------------------------------
```

In [13]:
```
columns = ['mushroom_is_edible', 'cap-shape', 'cap-surface', 'cap-color',
           'bruises','odor', 'gill-attachment', 'gill-spacing', 'gill-size',
           'gill-color','stalk-shape', 'stalk-root',
           'stalk-surface-above-ring','stalk-surface-below-ring',
           'stalk-color-above-ring','stalk-color-below-ring', 'veil-type',
           'veil-color', 'ring-number', 'ring-type', 'spore-print-color',
           'population', 'habitat'
          ]
```

In [14]:
```
df = pd.read_csv(data_filename, names=columns)
```

In [15]:
```
print(df.head())
```

```
  mushroom_is_edible cap-shape cap-surface cap-color bruises odor
0                  p         x           s         n       t    p  \
1                  e         x           s         y       t    a
2                  e         b           s         w       t    l
3                  p         x           y         w       t    p
4                  e         x           s         g       f    n

  gill-attachment gill-spacing gill-size gill-color  ...
0               f            c         n          k  ...  \
1               f            c         b          k  ...
2               f            c         b          n  ...
3               f            c         n          n  ...
4               f            w         b          k  ...

  stalk-surface-below-ring stalk-color-above-ring stalk-color-below-ring
0                        s                       w                       w  \
1                        s                       w                       w
2                        s                       w                       w
3                        s                       w                       w
4                        s                       w                       w

  veil-type veil-color ring-number ring-type spore-print-color population
0         p          w           o         p                 k          s  \
1         p          w           o         p                 n          n
2         p          w           o         p                 n          n
3         p          w           o         p                 k          s
4         p          w           o         e                 n          a

  habitat
0       u
1       g
2       m
3       u
4       g

[5 rows x 23 columns]
```

In [16]:
```
# check before encoding
df['mushroom_is_edible'].value_counts()
```

Out[16]:
```
mushroom_is_edible
e    4208
p    3916
Name: count, dtype: int64
```

In [50]:
```
print(df['mushroom_is_edible'].value_counts())
```

```
mushroom_is_edible
0    4208
1    3916
Name: count, dtype: int64
```

Now, we'll need to encode the letters as numbers

```
ref_enc = "https://stackoverflow.com/questions/24458645/" + "
"label-encoding-across-multiple-columns-in-scikit-learn"
```

Which reference was found in Chandra's stuff.

In [17]:
```
from collections import defaultdict

d = defaultdict(preprocessing.LabelEncoder)
df = df.apply(lambda x: d[x.name].fit_transform(x))
```

In [18]:
```
print(df.head())
```

```
   mushroom_is_edible  cap-shape  cap-surface  cap-color  bruises  odor
0                   1          5            2          4        1     6  \
1                   0          5            2          9        1     0
2                   0          0            2          8        1     3
3                   1          5            3          8        1     6
4                   0          5            2          3        0     5

   gill-attachment  gill-spacing  gill-size  gill-color  ...
0                1             0          1           4  ...  \
1                1             0          0           4  ...
2                1             0          0           5  ...
3                1             0          1           5  ...
4                1             1          0           4  ...

   stalk-surface-below-ring  stalk-color-above-ring  stalk-color-below-ring
0                         2                       7                       7  \
1                         2                       7                       7
2                         2                       7                       7
3                         2                       7                       7
4                         2                       7                       7

   veil-type  veil-color  ring-number  ring-type  spore-print-color
0          0           2            1          4                  2  \
1          0           2            1          4                  3
2          0           2            1          4                  3
3          0           2            1          4                  2
4          0           2            1          0                  3

   population  habitat
0           3        5
1           2        1
2           2        3
3           3        5
4           0        1

[5 rows x 23 columns]
```

In [19]:
```
# check after encoding
df['mushroom_is_edible'].value_counts()
```

Out[19]:
```
mushroom_is_edible
0    4208
1    3916
Name: count, dtype: int64
```

In [49]:
```
print(df['mushroom_is_edible'].value_counts())
```

```
mushroom_is_edible
0    4208
1    3916
Name: count, dtype: int64
```

In [20]:
```
# Nifty way to look at the data from Chandra
for key in d.keys():
    print(key, d[key].classes_)
```

```
mushroom_is_edible ['e' 'p']
cap-shape ['b' 'c' 'f' 'k' 's' 'x']
cap-surface ['f' 'g' 's' 'y']
cap-color ['b' 'c' 'e' 'g' 'n' 'p' 'r' 'u' 'w' 'y']
bruises ['f' 't']
odor ['a' 'c' 'f' 'l' 'm' 'n' 'p' 's' 'y']
gill-attachment ['a' 'f']
gill-spacing ['c' 'w']
gill-size ['b' 'n']
gill-color ['b' 'e' 'g' 'h' 'k' 'n' 'o' 'p' 'r' 'u' 'w' 'y']
stalk-shape ['e' 't']
stalk-root ['?' 'b' 'c' 'e' 'r']
stalk-surface-above-ring ['f' 'k' 's' 'y']
stalk-surface-below-ring ['f' 'k' 's' 'y']
stalk-color-above-ring ['b' 'c' 'e' 'g' 'n' 'o' 'p' 'w' 'y']
stalk-color-below-ring ['b' 'c' 'e' 'g' 'n' 'o' 'p' 'w' 'y']
veil-type ['p']
veil-color ['n' 'o' 'w' 'y']
ring-number ['n' 'o' 't']
ring-type ['e' 'f' 'l' 'n' 'p']
spore-print-color ['b' 'h' 'k' 'n' 'o' 'r' 'u' 'w' 'y']
population ['a' 'c' 'n' 's' 'v' 'y']
habitat ['d' 'g' 'l' 'm' 'p' 'u' 'w']
```

And we see the nice question mark (`'?'`) in the stalk-root column that we read about from the dataset site. Those are missing values.

In [21]:
```
#  What we, with xgboost, and SageMaker (with its xgboost)
#+ will need, _without_ column names, plus target as
#+ first column.
#  It seems sklearn likes to have the column names (?)
df.to_csv("mushroom_all_encoded.csv", index=False)
```

Let's follow `iris_data_preparation` for making our testing and validation sets.

Training and Validation Set

Target Variable as first column followed by input features:

mushroom_is_edible, cap-shape, cap-surface, cap-color, bruises,odor, gill-attachment, gill-spacing, gill-size, gill-color,stalk-shape, stalk-root, stalk-surface-above-ring,stalk-surface-below-ring, stalk-color-above-ring,stalk-color-below-ring, veil-type, veil-color, ring-number, ring-type, spore-print-color, population, habitat

Training and Validation files do not have a column header

(when feeding into sklearn or SageMaker)

In [22]:
```
#  Training   =  70% of the data
#+                 (of already-separated training;
#+                  nothing from test set)
#  Validation = 30% of the data
#+                 (of already-separating training;
#+                  nothing from test set)
#  We will randomize the order of the dataset entries

subj_prefix = 'mushroom'

fraction_for_training = 0.7
rnd_seed = 5

training_filename = subj_prefix + "_train.csv"
```

```
validation_filename = subj_prefix + "_validation.csv"
column_list_filename = subj_prefix + "_train_column_list.txt"
```

In [23]:
```
np.random.seed(rnd_seed)
l_shuffle = list(df.index)
np.random.shuffle(l_shuffle)
df = df.iloc[l_shuffle]
```

In [24]:
```
# numbers of entries (of rows) for each
rows = df.shape[0]
train = int(fraction_for_training * rows)
test = rows - train
```

In [25]:
```
rows, train, test
```

Out[25]:
```
(8124, 5686, 2438)
```

In [48]:
```
print(rows, train, test)
```

```
8124 5686 2438
```

In [26]:
```
# Write Training Set
df[:train].to_csv(training_filename,
                  index=False, header=False,
                  columns=columns
                 )
```

In [27]:
```
# Write Validation Set
df[train:].to_csv(validation_filename,
                  index=False, header=False,
                  columns=columns
                 )
```

In [28]:
```
# Write Column List
with open(column_list_filename, 'w') as f:
    f.write(','.join(columns))
##endof:  with open ... as f
```

In [29]:
```
# Let's see what we have
!ls -lah
```

```
total 1.3M
drwxrwxr-x  4 ec2-user ec2-user 4.0K Jul 31 02:59 .
drwxrwxr-x 10 ec2-user ec2-user 4.0K Jul 29 18:23 ..
drwxrwxr-x  2 ec2-user ec2-user 4.0K Jul 30 14:22 .ipynb_checkpoints
-rw-rw-r--  1 ec2-user ec2-user 365K Jul 31 03:00 mushroom_all.csv
-rw-rw-r--  1 ec2-user ec2-user 367K Jul 31 03:01 mushroom_all_encoded.csv
-rw-rw-r--  1 ec2-user ec2-user  22K Jul 31 02:59 Mushroom_Classification_dwb_try_2023-07-30.ipynb
-rw-rw-r--  1 ec2-user ec2-user  312 Jul 31 03:01 mushroom_train_column_list.txt
-rw-rw-r--  1 ec2-user ec2-user 257K Jul 31 03:01 mushroom_train.csv
drwxrwxr-x  3 ec2-user ec2-user 4.0K Jul 30 14:52 mushroom_unzipped
-rw-rw-r--  1 ec2-user ec2-user 110K Jul 31 03:01 mushroom_validation.csv
-rw-rw-r--  1 ec2-user ec2-user 139K Jul 31 02:59 mushroom.zip
```

Train a model with Mushroom data using XGBoost algorithm

Doing the training in the same notebook as data prep

Though the reading in of files shows that it could be done separately.

Model is trained with XGBoost, installed earlier in the notebook instance

In [30]:
```
column_list_file = "mushroom_train_column_list.txt"
train_file = "mushroom_train.csv"
validation_file = "mushroom_validation.csv"
```

In [31]:
```
columns = ""
with open(column_list_file, 'r') as tfh:
    columns = tfh.read().split(',')
##endof:  with open ... as tfh
```

In [32]:
```
columns
```

Out[32]:
```
['mushroom_is_edible',
 'cap-shape',
 'cap-surface',
 'cap-color',
 'bruises',
 'odor',
 'gill-attachment',
 'gill-spacing',
 'gill-size',
 'gill-color',
 'stalk-shape',
 'stalk-root',
 'stalk-surface-above-ring',
 'stalk-surface-below-ring',
 'stalk-color-above-ring',
 'stalk-color-below-ring',
 'veil-type',
 'veil-color',
 'ring-number',
 'ring-type',
 'spore-print-color',
 'population',
 'habitat']
```

In [33]:
```
print(columns)
```

```
['mushroom_is_edible', 'cap-shape', 'cap-surface', 'cap-color', 'bruises', 'odor', 'gill-attachment', 'gill-spacin
g', 'gill-size', 'gill-color', 'stalk-shape', 'stalk-root', 'stalk-surface-above-ring', 'stalk-surface-below-ring',
'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type', 'veil-color', 'ring-number', 'ring-type', 'spore-pr
int-color', 'population', 'habitat']
```

Actually, what's below isn't necessary at all in the mushroom dataset

<not-needed-for-mushrooms>

In [34]:
```
#labels=[0,1] # I'm almost positive this isn't necessary
classes = ['e', 'p']
le_2 = preprocessing.LabelEncoder()
le_2.fit(classes)
```

Out[34]:
```
LabelEncoder
LabelEncoder()
```

(From Chandra's notebook with the Iris dataset)

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.

Doing all the tab_completions with le_2 I can.

In [35]:
```
le_2.classes_
```

Out[35]:
```
array(['e', 'p'], dtype='<U1')
```

In [38]:
```
le_2.get_params()
```

Out[38]:
```
{}
```

</not-needed-for-mushrooms>

In [39]:
```
# Specify the column names as the file does not have a column header
df_train =      pd.read_csv(train_file,      names=columns)
df_validation = pd.read_csv(validation_file, names=columns)
```

In [40]:
```
print(df_train.head())
```

```
   mushroom_is_edible  cap-shape  cap-surface  cap-color  bruises  odor
0                   1          5            2          0        1     2  \
1                   1          3            3          4        0     7
2                   1          2            3          9        0     2
3                   0          0            3          8        1     0
4                   0          5            3          2        1     5

   gill-attachment  gill-spacing  gill-size  gill-color  ...
0                1             0          0           3  ...  \
1                1             0          1           0  ...
2                1             0          0           2  ...
3                1             0          0          10  ...
4                1             0          0           7  ...

   stalk-surface-below-ring  stalk-color-above-ring  stalk-color-below-ring
0                         0                       7                       7  \
1                         1                       6                       6
2                         1                       6                       6
3                         2                       7                       7
4                         2                       6                       3

   veil-type  veil-color  ring-number  ring-type  spore-print-color
0          0           2            1          4                  1  \
1          0           2            1          0                  7
2          0           2            1          2                  1
3          0           2            1          4                  2
4          0           2            1          4                  2

   population  habitat
0           3        1
1           4        0
2           4        1
3           2        3
4           4        0

[5 rows x 23 columns]
```

In [41]:
```
print(df_validation.head())
```

```
   mushroom_is_edible  cap-shape  cap-surface  cap-color  bruises  odor      \
0                   0          2            0          8        0     5  \
1                   0          5            2          3        0     5
2                   0          5            0          8        0     5
3                   0          5            2          4        0     5
4                   0          5            3          3        1     5

   gill-attachment  gill-spacing  gill-size  gill-color  ...
0                1             1          0           7  ...  \
1                1             1          0           5  ...
2                1             1          0           4  ...
3                0             0          0           5  ...
4                1             0          0           5  ...

   stalk-surface-below-ring  stalk-color-above-ring  stalk-color-below-ring
0                         2                       7                       7  \
1                         2                       7                       7
2                         2                       7                       7
3                         2                       5                       5
4                         2                       6                       6

   veil-type  veil-color  ring-number  ring-type  spore-print-color
0          0           2            1          0                  2  \
1          0           2            1          0                  3
2          0           2            1          0                  2
3          0           1            1          4                  0
4          0           2            1          4                  3

   population  habitat
0           0        1
1           3        1
2           0        1
3           4        2
4           5        0

[5 rows x 23 columns]
```

Here, we'll actually split up the dataframes as needed and train with the classifier

START: Optional inspection of how the dataframe gets split with iloc

In [42]:
```
print(df_train.head())
```

```
   mushroom_is_edible  cap-shape  cap-surface  cap-color  bruises  odor
0                   1          5            2          0        1     2  \
1                   1          3            3          4        0     7
2                   1          2            3          9        0     2
3                   0          0            3          8        1     0
4                   0          5            3          2        1     5

   gill-attachment  gill-spacing  gill-size  gill-color  ...
0                1             0          0           3  ...  \
1                1             0          1           0  ...
2                1             0          0           2  ...
3                1             0          0          10  ...
4                1             0          0           7  ...

   stalk-surface-below-ring  stalk-color-above-ring  stalk-color-below-ring
0                         0                       7                       7  \
1                         1                       6                       6
2                         1                       6                       6
3                         2                       7                       7
4                         2                       6                       3

   veil-type  veil-color  ring-number  ring-type  spore-print-color
0          0           2            1          4                  1  \
1          0           2            1          0                  7
2          0           2            1          2                  1
3          0           2            1          4                  2
4          0           2            1          4                  2

   population  habitat
0           3        1
1           4        0
2           4        1
3           2        3
4           4        0

[5 rows x 23 columns]
```

In [43]:
```
print(df_train.iloc[:,1:].head()) # All rows, columns 1 and on (zero-indexed)
```

```
   cap-shape  cap-surface  cap-color  bruises  odor  gill-attachment
0          5            2          0        1     2                1  \
1          3            3          4        0     7                1
2          2            3          9        0     2                1
3          0            3          8        1     0                1
4          5            3          2        1     5                1

   gill-spacing  gill-size  gill-color  stalk-shape  ...
0             0          0           3            1  ...  \
1             0          1           0            1  ...
2             0          0           2            0  ...
3             0          0          10            0  ...
4             0          0           7            1  ...

   stalk-surface-below-ring  stalk-color-above-ring  stalk-color-below-ring
0                         0                       7                       7  \
1                         1                       6                       6
2                         1                       6                       6
3                         2                       7                       7
4                         2                       6                       3

   veil-type  veil-color  ring-number  ring-type  spore-print-color
0          0           2            1          4                  1  \
1          0           2            1          0                  7
2          0           2            1          2                  1
3          0           2            1          4                  2
4          0           2            1          4                  2

   population  habitat
0           3        1
1           4        0
2           4        1
3           2        3
4           4        0

[5 rows x 22 columns]
```

In [44]:
```
print(df_train.iloc[:, 0].head()) #  All rows, column 0
                                  #+ The column header
                                  #+ is at the bottom.
```

```
0    1
1    1
2    1
3    0
4    0
Name: mushroom_is_edible, dtype: int64
```

In [45]:
```
df_train.iloc[:,0].ravel() # takes column zero and switches it to an array.
```

Out[45]:
```
array([1, 1, 1, ..., 0, 0, 0])
```

In [46]:
```
print(df_train.iloc[:,0].ravel()) # takes column zero and switches it to an array.
```

```
[1 1 1 ... 0 0 0]
```

ENDOF: Optional inspection of how the dataframe gets split with iloc

In [56]:
```
X_train = df_train.iloc[:, 1:]
y_train = df_train.iloc[:,0].ravel()

X_validation = df_validation.iloc[:, 1:]
y_validation = df_validation.iloc[:,0].ravel()
```

In [57]:
```
# Launch a classifier
# XGBoost Training Parameter Reference:
#   https://github.com/dmlc/xgboost/blob/master/doc/parameter.md
#classifier = xgb.XGBClassifier(objective='binary:logistic'
#                               n_estimators=50)
classifier = xgb.XGBClassifier(objective='binary:logistic')
```

In [58]:
```
classifier
```

Out[58]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [59]:

```
print(classifier)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

In [60]:

```
classifier.fit(X_train,
               y_train,
               eval_set = [(X_train, y_train), (X_validation, y_validation)],
               eval_metric=['logloss'])
```

```
[0]    validation_0-logloss:0.44414    validation_1-logloss:0.44445
[1]    validation_0-logloss:0.30564    validation_1-logloss:0.30641
[2]    validation_0-logloss:0.21464    validation_1-logloss:0.21545
[3]    validation_0-logloss:0.15405    validation_1-logloss:0.15500
[4]    validation_0-logloss:0.11200    validation_1-logloss:0.11277
[5]    validation_0-logloss:0.08241    validation_1-logloss:0.08331
[6]    validation_0-logloss:0.06134    validation_1-logloss:0.06207
[7]    validation_0-logloss:0.04607    validation_1-logloss:0.04666
[8]    validation_0-logloss:0.03465    validation_1-logloss:0.03513
[9]    validation_0-logloss:0.02636    validation_1-logloss:0.02688
[10]   validation_0-logloss:0.02031    validation_1-logloss:0.02076
[11]   validation_0-logloss:0.01528    validation_1-logloss:0.01563
[12]   validation_0-logloss:0.01158    validation_1-logloss:0.01186
[13]   validation_0-logloss:0.00891    validation_1-logloss:0.00912
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/xgboost/sklearn.py:835: UserWarning: `eval_metric
` in `fit` method is deprecated for better compatibility with scikit-learn, use `eval_metric` in constructor or`set_
params` instead.
  warnings.warn(
```

```
[14]    validation_0-logloss:0.00703    validation_1-logloss:0.00722
[15]    validation_0-logloss:0.00566    validation_1-logloss:0.00582
[16]    validation_0-logloss:0.00445    validation_1-logloss:0.00459
[17]    validation_0-logloss:0.00372    validation_1-logloss:0.00385
[18]    validation_0-logloss:0.00302    validation_1-logloss:0.00314
[19]    validation_0-logloss:0.00261    validation_1-logloss:0.00272
[20]    validation_0-logloss:0.00227    validation_1-logloss:0.00237
[21]    validation_0-logloss:0.00198    validation_1-logloss:0.00206
[22]    validation_0-logloss:0.00175    validation_1-logloss:0.00183
[23]    validation_0-logloss:0.00165    validation_1-logloss:0.00173
[24]    validation_0-logloss:0.00150    validation_1-logloss:0.00158
[25]    validation_0-logloss:0.00139    validation_1-logloss:0.00147
[26]    validation_0-logloss:0.00132    validation_1-logloss:0.00140
[27]    validation_0-logloss:0.00124    validation_1-logloss:0.00133
[28]    validation_0-logloss:0.00120    validation_1-logloss:0.00129
[29]    validation_0-logloss:0.00116    validation_1-logloss:0.00126
[30]    validation_0-logloss:0.00112    validation_1-logloss:0.00121
[31]    validation_0-logloss:0.00109    validation_1-logloss:0.00119
[32]    validation_0-logloss:0.00106    validation_1-logloss:0.00116
[33]    validation_0-logloss:0.00104    validation_1-logloss:0.00114
[34]    validation_0-logloss:0.00102    validation_1-logloss:0.00112
[35]    validation_0-logloss:0.00100    validation_1-logloss:0.00110
[36]    validation_0-logloss:0.00098    validation_1-logloss:0.00108
[37]    validation_0-logloss:0.00096    validation_1-logloss:0.00106
[38]    validation_0-logloss:0.00095    validation_1-logloss:0.00104
[39]    validation_0-logloss:0.00093    validation_1-logloss:0.00103
[40]    validation_0-logloss:0.00092    validation_1-logloss:0.00101
[41]    validation_0-logloss:0.00091    validation_1-logloss:0.00100
[42]    validation_0-logloss:0.00089    validation_1-logloss:0.00098
[43]    validation_0-logloss:0.00088    validation_1-logloss:0.00097
[44]    validation_0-logloss:0.00087    validation_1-logloss:0.00096
[45]    validation_0-logloss:0.00086    validation_1-logloss:0.00095
[46]    validation_0-logloss:0.00085    validation_1-logloss:0.00093
[47]    validation_0-logloss:0.00084    validation_1-logloss:0.00092
[48]    validation_0-logloss:0.00083    validation_1-logloss:0.00091
[49]    validation_0-logloss:0.00081    validation_1-logloss:0.00090
[50]    validation_0-logloss:0.00081    validation_1-logloss:0.00089
[51]    validation_0-logloss:0.00080    validation_1-logloss:0.00088
[52]    validation_0-logloss:0.00079    validation_1-logloss:0.00087
[53]    validation_0-logloss:0.00079    validation_1-logloss:0.00087
[54]    validation_0-logloss:0.00078    validation_1-logloss:0.00086
[55]    validation_0-logloss:0.00077    validation_1-logloss:0.00085
```

```
[56]    validation_0-logloss:0.00076    validation_1-logloss:0.00085
[57]    validation_0-logloss:0.00076    validation_1-logloss:0.00084
[58]    validation_0-logloss:0.00075    validation_1-logloss:0.00083
[59]    validation_0-logloss:0.00075    validation_1-logloss:0.00083
[60]    validation_0-logloss:0.00074    validation_1-logloss:0.00082
[61]    validation_0-logloss:0.00074    validation_1-logloss:0.00082
[62]    validation_0-logloss:0.00073    validation_1-logloss:0.00081
[63]    validation_0-logloss:0.00073    validation_1-logloss:0.00081
[64]    validation_0-logloss:0.00073    validation_1-logloss:0.00080
[65]    validation_0-logloss:0.00072    validation_1-logloss:0.00080
[66]    validation_0-logloss:0.00072    validation_1-logloss:0.00079
[67]    validation_0-logloss:0.00071    validation_1-logloss:0.00079
[68]    validation_0-logloss:0.00071    validation_1-logloss:0.00078
[69]    validation_0-logloss:0.00071    validation_1-logloss:0.00078
[70]    validation_0-logloss:0.00070    validation_1-logloss:0.00077
[71]    validation_0-logloss:0.00070    validation_1-logloss:0.00077
[72]    validation_0-logloss:0.00069    validation_1-logloss:0.00076
[73]    validation_0-logloss:0.00069    validation_1-logloss:0.00076
[74]    validation_0-logloss:0.00069    validation_1-logloss:0.00075
[75]    validation_0-logloss:0.00069    validation_1-logloss:0.00075
[76]    validation_0-logloss:0.00068    validation_1-logloss:0.00075
[77]    validation_0-logloss:0.00068    validation_1-logloss:0.00075
[78]    validation_0-logloss:0.00068    validation_1-logloss:0.00074
[79]    validation_0-logloss:0.00067    validation_1-logloss:0.00074
[80]    validation_0-logloss:0.00067    validation_1-logloss:0.00073
[81]    validation_0-logloss:0.00067    validation_1-logloss:0.00073
[82]    validation_0-logloss:0.00066    validation_1-logloss:0.00073
[83]    validation_0-logloss:0.00066    validation_1-logloss:0.00072
[84]    validation_0-logloss:0.00065    validation_1-logloss:0.00072
[85]    validation_0-logloss:0.00065    validation_1-logloss:0.00072
[86]    validation_0-logloss:0.00065    validation_1-logloss:0.00071
[87]    validation_0-logloss:0.00065    validation_1-logloss:0.00071
[88]    validation_0-logloss:0.00065    validation_1-logloss:0.00071
[89]    validation_0-logloss:0.00064    validation_1-logloss:0.00071
[90]    validation_0-logloss:0.00064    validation_1-logloss:0.00071
[91]    validation_0-logloss:0.00064    validation_1-logloss:0.00070
[92]    validation_0-logloss:0.00064    validation_1-logloss:0.00070
[93]    validation_0-logloss:0.00063    validation_1-logloss:0.00070
[94]    validation_0-logloss:0.00063    validation_1-logloss:0.00069
[95]    validation_0-logloss:0.00063    validation_1-logloss:0.00069
[96]    validation_0-logloss:0.00063    validation_1-logloss:0.00069
[97]    validation_0-logloss:0.00062    validation_1-logloss:0.00069
```

```
[98]    validation_0-logloss:0.00062    validation_1-logloss:0.00068
[99]    validation_0-logloss:0.00062    validation_1-logloss:0.00068
```

Out[60]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [61]:
```
eval_result = classifier.evals_result()
```

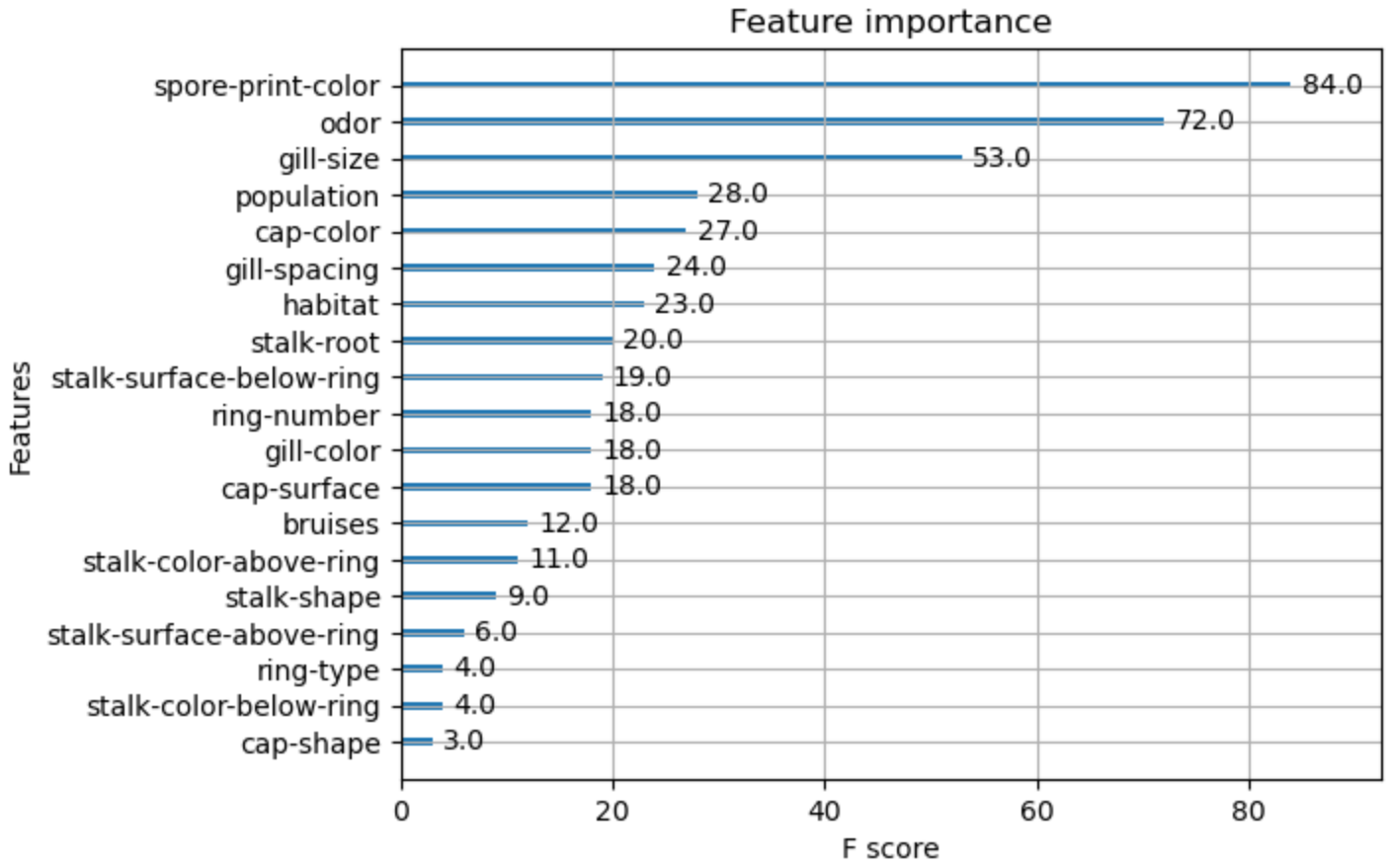
In [62]:
```
training_rounds = range(len(eval_result['validation_0']['logloss']))
```

In [63]:
```
print(training_rounds)
```

```
range(0, 100)
```

In [64]:
```
plt.scatter(x=training_rounds,
            y=eval_result['validation_0']['logloss'],
            label="Training Error")
plt.scatter(x=training_rounds,
            y=eval_result['validation_1']['logloss'],
            label="Validation Error")
plt.grid(True)
plt.xlabel("Iterations")
plt.ylabel("LogLoss")
plt.title("Training Vs Validation Error")
plt.legend()
plt.show()
```

In [65]:
```
xgb.plot_importance(classifier)
plt.show()
```



In [66]:
```
df = pd.read_csv(validation_file, names=columns)
```

In [67]:
```
print(df.head())
```

```
   mushroom_is_edible  cap-shape  cap-surface  cap-color  bruises  odor      \
0                   0          2            0          8        0     5  \
1                   0          5            2          3        0     5
2                   0          5            0          8        0     5
3                   0          5            2          4        0     5
4                   0          5            3          3        1     5

   gill-attachment  gill-spacing  gill-size  gill-color  ...
0                1             1          0           7  ...  \
1                1             1          0           5  ...
2                1             1          0           4  ...
3                0             0          0           5  ...
4                1             0          0           5  ...

   stalk-surface-below-ring  stalk-color-above-ring  stalk-color-below-ring
0                         2                       7                       7  \
1                         2                       7                       7
2                         2                       7                       7
3                         2                       5                       5
4                         2                       6                       6

   veil-type  veil-color  ring-number  ring-type  spore-print-color
0          0           2            1          0                  2  \
1          0           2            1          0                  3
2          0           2            1          0                  2
3          0           1            1          4                  0
4          0           2            1          4                  3

   population  habitat
0           0        1
1           3        1
2           0        1
3           4        2
4           5        0

[5 rows x 23 columns]
```

Prediction Time

In [68]:
```
X_test = df.iloc[:, 1:]
```

In [69]:
```
result = classifier.predict(X_test)
```

In [72]:
```
# Let's look at a few of the predictions
result[:5]  # shoot, all five are poisonous
```

Out[72]:
```
array([0, 0, 0, 0, 0])
```

In [74]:
```
# Does the end look any better?
result[-5:]  # all right, some edible ones
```

Out[74]:
```
array([0, 1, 1, 0, 0])
```

In [75]:
```
df['predicted_class'] = result
```

In [76]:
```
print(df.head())
```

```
   mushroom_is_edible  cap-shape  cap-surface  cap-color  bruises  odor      \
0                   0          2            0          8        0     5  \
1                   0          5            2          3        0     5
2                   0          5            0          8        0     5
3                   0          5            2          4        0     5
4                   0          5            3          3        1     5

   gill-attachment  gill-spacing  gill-size  gill-color  ...
0                1             1          0           7  ...  \
1                1             1          0           5  ...
2                1             1          0           4  ...
3                0             0          0           5  ...
4                1             0          0           5  ...

   stalk-color-above-ring  stalk-color-below-ring  veil-type  veil-color      \
0                       7                       7          0           2  \
1                       7                       7          0           2
2                       7                       7          0           2
3                       5                       5          0           1
4                       6                       6          0           2

   ring-number  ring-type  spore-print-color  population  habitat      \
0            1          0                  2           0        1  \
1            1          0                  3           3        1
2            1          0                  2           0        1
3            1          4                  0           4        2
4            1          4                  3           5        0

   predicted_class
0                0
1                0
2                0
3                0
4                0

[5 rows x 24 columns]
```

In [77]:
```
print(df.tail())
```

```
      mushroom_is_edible  cap-shape  cap-surface  cap-color  bruises  odor  \
2433                   0          5            3          2        1     5
2434                   1          2            3          8        1     6
2435                   1          2            0          9        0     2
2436                   0          5            3          4        1     5
2437                   0          5            3          3        1     5

      gill-attachment  gill-spacing  gill-size  gill-color  ...  \
2433                1             0          0           5  ...
2434                1             0          1           4  ...
2435                1             0          0           2  ...
2436                1             0          0          10  ...
2437                1             0          0           9  ...

      stalk-color-above-ring  stalk-color-below-ring  veil-type  veil-color  \
2433                       6                       6          0           2
2434                       7                       7          0           2
2435                       0                       0          0           2
2436                       7                       7          0           2
2437                       3                       3          0           2

      ring-number  ring-type  spore-print-color  population  habitat  \
2433            1          4                  2           4        0
2434            1          4                  2           4        1
2435            1          2                  1           4        1
2436            1          4                  2           4        0
2437            1          4                  3           4        0

      predicted_class
2433                0
2434                1
2435                1
2436                0
2437                0

[5 rows x 24 columns]
```

In [78]:
```
df.mushroom_is_edible.value_counts()
```

Out[78]:
```
mushroom_is_edible
0    1243
1    1195
Name: count, dtype: int64
```

In [79]:
```
print(df['mushroom_is_edible'].value_counts())
```

```
mushroom_is_edible
0    1243
1    1195
Name: count, dtype: int64
```

In [80]:
```
df.predicted_class.value_counts()
```

Out[80]:
```
predicted_class
0    1243
1    1195
Name: count, dtype: int64
```

In [81]:
```
print(df['predicted_class'].value_counts())
```

```
predicted_class
0    1243
1    1195
Name: count, dtype: int64
```

That looks uh pretty-pretty good.

Binary Classifier Metrics

I'm just following the patterns from the solution, rather than an earlier notebook.

In [82]:
```
# Reference: https://scikit-learn.org/stable/modules/model_evaluation.html
# Explicitly stating labels. Pass=1, Fail=0
def true_positive(y_true, y_pred):
    return confusion_matrix(y_true, y_pred, labels=[1, 0])[0, 0]
                        # positions in  confusion matrix -^--^-
##endof:  true_positive(y_true, y_pred)

def true_negative(y_true, y_pred):
```

```
    return confusion_matrix(y_true, y_pred, labels=[1, 0])[1, 1]
##endof:  true_negative(y_true, y_pred)

def false_positive(y_true, y_pred):
    return confusion_matrix(y_true, y_pred, labels=[1, 0])[1, 0]
##endof:  false_positive(y_true, y_pred)

def false_negative(y_true, y_pred):
    return confusion_matrix(y_true, y_pred, labels=[1, 0])[0, 1]
```

In [83]:

```
# Compute Binary Classifier Metrics
# Returns a dictionary {"MetricName":Value,...}

def binary_classifier_metrics(y_true, y_pred):
    metrics = {}

    # References:
    #  https://docs.aws.amazon.com/machine-learning/latest/dg/binary-classification.html
    #  https://en.wikipedia.org/wiki/Confusion_matrix

    # Definition:
    # true positive = tp = how many samples were correctly classified as positive (count)
    # true negative = tn = how many samples were correctly classified as negative (count)
    # false positive = fp = how many negative samples were mis-classified as positive (count)
    # false_negative = fn = how many positive samples were mis-classified as negative (count)

    # positive = number of positive samples (count)
    #          = true positive + false negative
    # negative = number of negative samples (count)
    #          = true negative + false positive

    tp = true_positive(y_true, y_pred)
    tn = true_negative(y_true, y_pred)
    fp = false_positive(y_true, y_pred)
    fn = false_negative(y_true, y_pred)

    positive = tp + fn
    negative = tn + fp

    metrics['TruePositive'] = tp
    metrics['TrueNegative'] = tn
    metrics['FalsePositive'] = fp
```

```
    metrics['FalseNegative'] = fn

    metrics['Positive'] = positive
    metrics['Negative'] = negative

    # True Positive Rate (TPR, Recall) = true positive/positive
    # How many positives were correctly classified? (fraction)
    # Recall value closer to 1 is better. closer to 0 is worse
    if tp == 0:
        recall = 0
    else:
        recall = tp/positive

    metrics['Recall'] = recall

    # True Negative Rate = True Negative/negative
    # How many negatives were correctly classified? (fraction)
    # True Negative Rate value closer to 1 is better. closer to 0 is worse
    if tn == 0:
        tnr = 0
    else:
        tnr = tn/(negative)
    metrics['TrueNegativeRate'] = tnr

    # Precision = True Positive/(True Positive + False Positive)
    # How many positives classified by the algorithm are really positives? (fraction)
    # Precision value closer to 1 is better. closer to 0 is worse
    if tp == 0:
        precision = 0
    else:
        precision = tp/(tp + fp)
    metrics['Precision'] = precision

    # Accuracy = (True Positive + True Negative)/(total positive + total negative)
    # How many positives and negatives were correctly classified? (fraction)
    # Accuracy value closer to 1 is better. closer to 0 is worse
    accuracy = (tp + tn)/(positive + negative)
    metrics['Accuracy'] = accuracy

    # False Positive Rate (FPR, False Alarm) = False Positive/(total negative)
    # How many negatives were mis-classified as positives (fraction)
    # False Positive Rate value closer to 0 is better. closer to 1 is worse
```

```
    if fp == 0:
        fpr = 0
    else:
        fpr = fp/(negative)
    metrics['FalsePositiveRate'] = fpr

    # False Negative Rate (FNR, Misses) = False Negative/(total Positive)
    # How many positives were mis-classified as negative (fraction)
    # False Negative Rate value closer to 0 is better. closer to 1 is worse
    fnr = fn/(positive)
    metrics['FalseNegativeRate'] = fnr

    # F1 Score = harmonic mean of Precision and Recall
    # F1 Score closer to 1 is better. Closer to 0 is worse.
    if precision == 0 or recall == 0:
        f1 = 0
    else:
        f1 = 2*precision*recall/(precision+recall)

    metrics['F1'] = f1

    return metrics
```

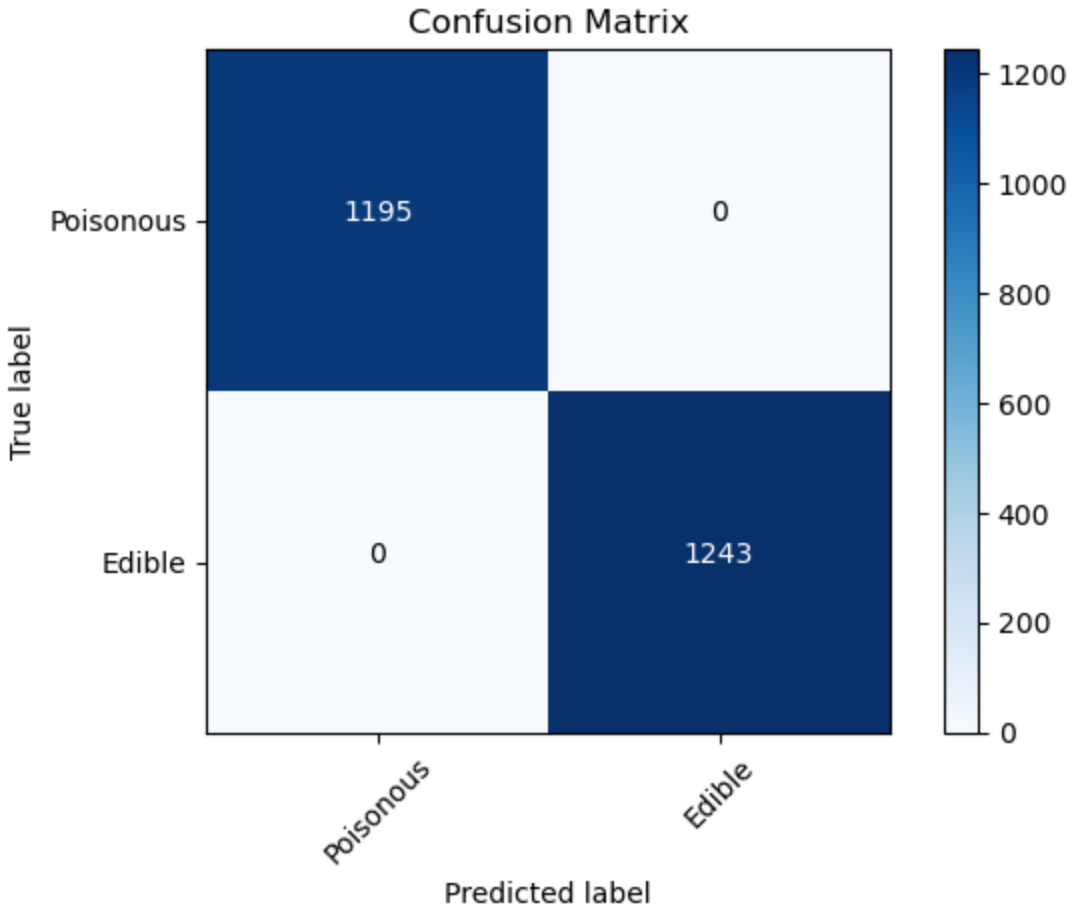
In [84]:

```
# Reference:
# https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        #print("Normalized confusion matrix")
    #else:
    #    print('Confusion matrix, without normalization')

    #print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

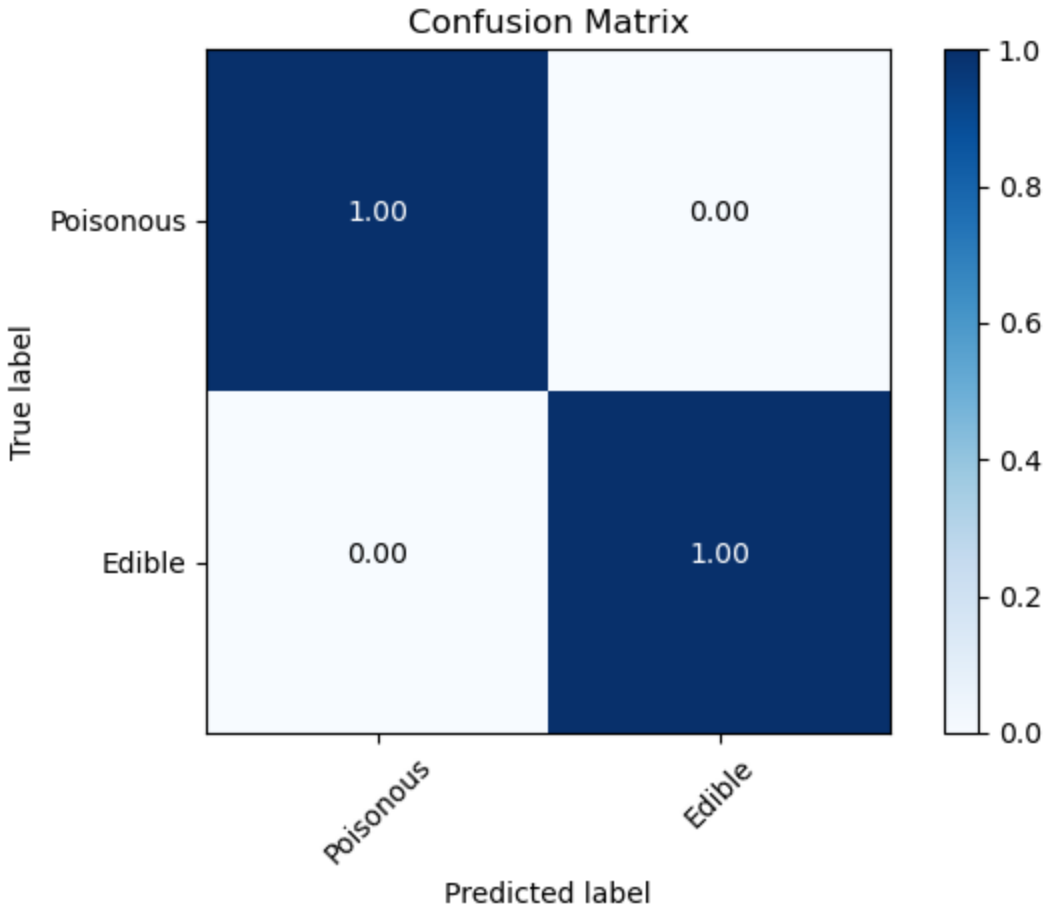
```
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel("True label")
    plt.xlabel("Predicted label")
    plt.tight_layout()
```

In [86]:
```
# Compute confusion matrix
#{0:'edible',1:'poisonous'}) <-- WHAT!?!?!
cnf_matrix = confusion_matrix(df['mushroom_is_edible'],
                              df['predicted_class'],
                              labels=[1, 0])
```

In [87]:
```
# Plot confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=["Poisonous", "Edible"],
                      title="Confusion Matrix")
```



In [88]:
```
# Plot confusion matrix - fractions
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=["Poisonous", "Edible"],
                      title="Confusion Matrix",
                      normalize=True)
```



In [90]:
```
metrics = [binary_classifier_metrics(df['mushroom_is_edible'], df['predicted_class'])]
df_metrics=pd.DataFrame.from_dict(metrics)
df_metrics.index = ['Model']
```

In [91]:
```
df_metrics
```

Out[91]:

	TruePositive	TrueNegative	FalsePositive	FalseNegative	Positive	Negative	Recall	TrueNegativeRate	Precision	Accuracy	FalsePo
Model	1195	1243	0	0	1195	1243	1.0	1.0	1.0	1.0	

In [92]:
```
print(df_metrics)
```

```
       TruePositive  TrueNegative  FalsePositive  FalseNegative  Positive
Model          1195          1243              0              0      1195  \

       Negative  Recall  TrueNegativeRate  Precision  Accuracy
Model      1243     1.0               1.0        1.0       1.0  \

       FalsePositiveRate  FalseNegativeRate   F1
Model                  0                0.0  1.0
```

In [93]:
```
print('Counts')
print(df_metrics[['TruePositive',
                  'FalseNegative',
                  'FalsePositive',
                  'TrueNegative',]].round(2))
print()
print('Fractions')
print(df_metrics[['Recall',
                  'FalseNegativeRate',
                  'FalsePositiveRate',
                  'TrueNegativeRate',]].round(2))
print()

print(df_metrics[['Precision',
                  'Accuracy',
                  'F1']].round(2))
```

```
Counts
       TruePositive  FalseNegative  FalsePositive  TrueNegative
Model          1195              0              0          1243

Fractions
       Recall  FalseNegativeRate  FalsePositiveRate  TrueNegativeRate
Model     1.0                0.0                  0               1.0

       Precision  Accuracy   F1
Model        1.0       1.0  1.0
```

In [95]:
```
print(classification_report(df['mushroom_is_edible'],
                            df['predicted_class'],
                            labels=[1, 0],
                            target_names=['Poisonous','Edible']
```

```
                        )
    )
```

```
              precision    recall  f1-score   support

   Poisonous       1.00      1.00      1.00      1195
      Edible       1.00      1.00      1.00      1243

    accuracy                           1.00      2438
   macro avg       1.00      1.00      1.00      2438
weighted avg       1.00      1.00      1.00      2438
```

In [ ]: