```
#!/usr/bin/env python
# coding: utf-8

# ### Libraries

# In[47]:


import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import category_encoders as ce
from sklearn.metrics import confusion_matrix
from sklearn import tree
import warnings

warnings.filterwarnings('ignore')


# In[24]:


conda install -c conda-forge category_encoders


# ### Loading the data

# In[52]:


df = pd.read_excel("/Users/snehaltikone/Spring_2022/DataMining/
Decision Tree Project/mushrooms.xlsx")


# In[53]:


print(df.info());


# In[55]:


df.head(10)
```

```
# ### Checking for NULL,NA or duplicate values

# In[4]:


print("Null Data:\n",df.isnull().sum());
print("\nNA Data:\n",df.isna().sum());


# In[5]:


print(df.duplicated())


# In[58]:


df.dtypes


# In[8]:


df.describe()


# In[9]:


df.columns


# In[12]:


for col in df:
  print(df[col].value_counts())


# In[11]:


df.head(10)


# ### Summary
#
# 1. There are 23 columns and 8124 rows in the dataset
# 2. All the variables are categorical.
```

```
# 3. There are no null values in the dataset.
# 4. No duplicate rows.
# 5. The class is the target variable while other are the features.
# 6. The class can have either values
#     p - mushroom is poisonous
#     e - mushroom is edible
#

# ## Train and Test Dataset

# In[14]:


X = df.drop(['class'], axis=1) #Features

y = df['class'] #Target Variable


# In[15]:


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.35, random_state = 42)


# In[16]:


X_train.shape, X_test.shape


# In[17]:


y_test.shape, y_test.shape


# In[29]:


# encode variables with ordinal encoding

encoder = ce.OrdinalEncoder(cols=['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'])
```

```
X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)


# In[32]:


#Instantiating the decision tree using the entropy criterion

des_tree = DecisionTreeClassifier(criterion='entropy', max_depth=4,
random_state=0)

#Fitting the model
des_tree.fit(X_train,y_train)


# #### Predicting the results

# In[34]:


y_pred = des_tree.predict(X_test)


# In[37]:


print('Accuracy Score: {0:0.4f}'. format(accuracy_score(y_test,
y_pred)))


# ### Comparing the train and test results

# In[38]:


y_pred_train = des_tree.predict(X_train)

y_pred_train


# In[39]:


# print the scores on training and test set

print('Training set score: {:.4f}'.format(des_tree.score(X_train,
```

```
y_train)))

print('Test set score: {:.4f}'.format(des_tree.score(X_test, y_test)))


# In[49]:


plt.figure(figsize=(15,12))
tree.plot_tree(des_tree,fontsize=10);


# In[48]:


# Print the Confusion Matrix and slice it into four pieces
con_mat = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', con_mat)


# In[76]:


sns.heatmap(con_mat/np.sum(con_mat), annot=True,
fmt='.2%').set_title("Confusion Matrix")


# In[77]:


y_pred


# In[79]:


np.unique(y_pred,return_counts=True)


# In[80]:


np.unique(y_test,return_counts=True)


# In[ ]:
```