Blog -1

Toxicity Detection of Mushrooms Using Machine Learning

Batch Id :1829

1. Problem Definition:

Mushroom is one of the best nutritional food with high proteins, vitamins and minerals. It contains antioxidants that prevent people from heart disease and cancer. They provide carbohydrates of high quality enhancing the human health. Mushrooms are considered as substitute for meat and its nutritional value is comparable to several vegetables. The total production of mushroom world-wide is about 40 million tones majorly contributed by China, USA, Europe and Canada. Furthermore, Mushroom production is a lucrative and profitable business for cottage industry and providing mass employment in many developing countries

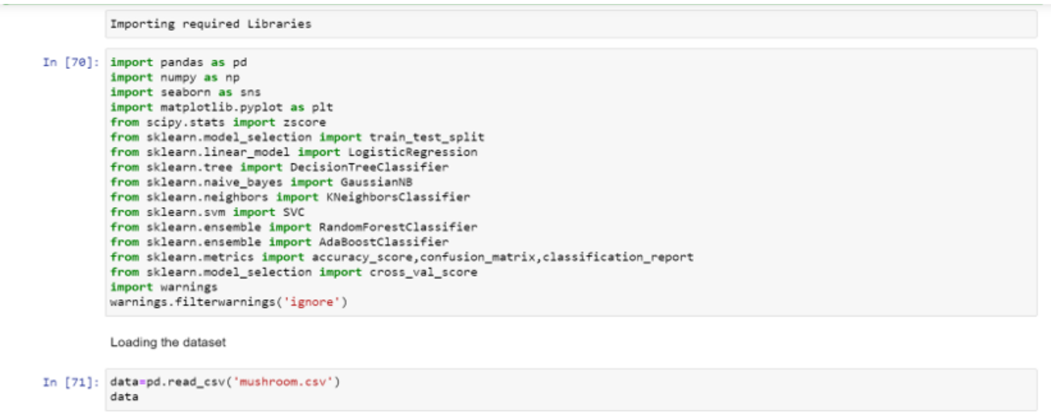
However, On the other hand, some mushrooms are toxic and dangerous if we eat them. Therefore, it is a pertinent to differentiate, the edible and poisonous mushrooms. There are 45000 species of mushroom found to be existing world-wide, but the number of species of edible mushrooms is only 2000. Identifying edible or poisonous mushroom through the naked eye is quite difficult. Because maximum poisonous mushrooms look like edible mushroom owing to color and shape. Even there is no easy rule to identify toxicity in mushrooms using machine learning methods that work for all types of data. Our objective is to find an efficient method for identifying toxicity of mushroom with the highest accuracy and lowest error rate.

2. Exploratory Data Analysis (EDA)

At first for classifying the mushrooms, a dataset containing 8124 instances and 23 variables of mushroom was obtained from UCI Machine Learning Repository. In this part, we will try to gain insights and a get familiar with the data.

2.1 Importing Libraries and Dataset

First import the required libraries like NumPy, pandas, matplotlib, and seaborn in our notebook. Then load the data set from CSV format and convert it into Data Frame.



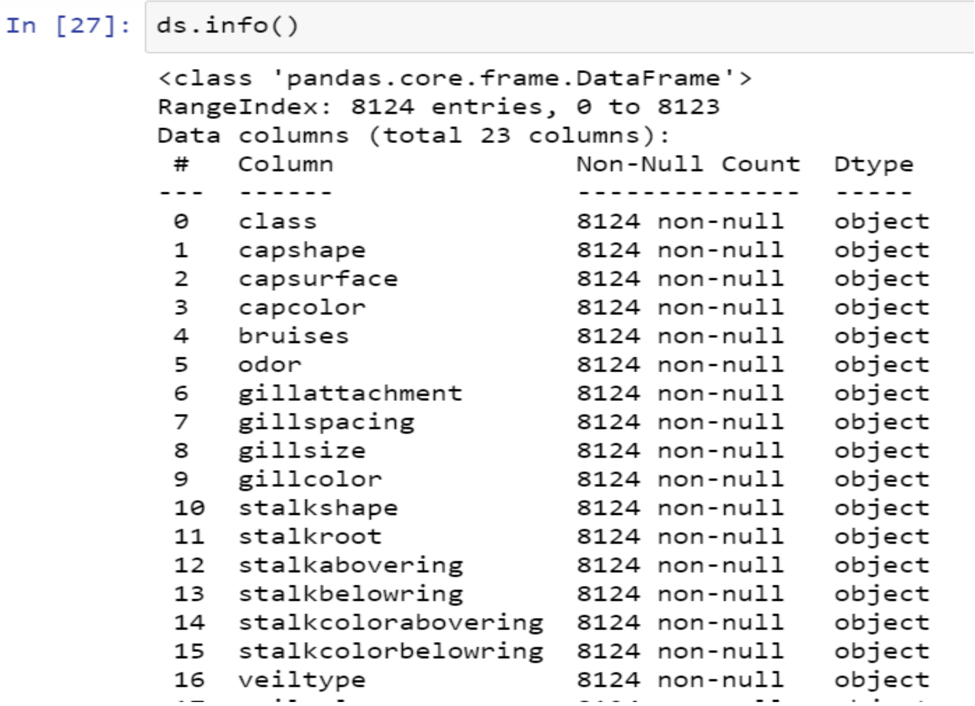
Importing required Libraries

```
In [70]: import pandas as pd
         import numpy as np
         import seaborn as sns
         import matplotlib.pyplot as plt
         from scipy.stats import zscore
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.naive_bayes import GaussianNB
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.svm import SVC
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.ensemble import AdaBoostClassifier
         from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
         from sklearn.model_selection import cross_val_score
         import warnings
         warnings.filterwarnings('ignore')
```

Loading the dataset

```
In [71]: data=pd.read_csv('mushroom.csv')
         data
```

2.2 Attribute Information:



```
In [27]: ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   class                8124 non-null   object
 1   capshape             8124 non-null   object
 2   capsurface           8124 non-null   object
 3   capcolor             8124 non-null   object
 4   bruises              8124 non-null   object
 5   odor                 8124 non-null   object
 6   gillattachment       8124 non-null   object
 7   gillspacing          8124 non-null   object
 8   gillsize             8124 non-null   object
 9   gillcolor            8124 non-null   object
 10  stalkshape           8124 non-null   object
 11  stalkroot            8124 non-null   object
 12  stalkabovering       8124 non-null   object
 13  stalkbelowring       8124 non-null   object
 14  stalkcolorabovering  8124 non-null   object
 15  stalkcolorbelowring  8124 non-null   object
 16  veiltype             8124 non-null   object
```

1. **cap-shape**: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s

2. **cap-surface**: fibrous=f, grooves=g, scaly=y, smooth=s

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

4. **bruises** ?: bruises=t, no=f

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

6. **gill-attachment:** attached=a,descending=d,free=f,notched=n

7. **gill-spacing**: close=c,crowded=w,distant=d

8. **gill-size:** broad=b,narrow=n

9.**gillcolor:**black=k,brown=n,buff=b,chocolate=h,gray=g,green=r,orange=o,pink=p,purple =u,red=e, white=w,yellow=y

10. **stalk-shape**: enlarging=e,tapering=t

11. **stalk-root**: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?

12. **stalk-surface-above-ring:** fibrous=f,scaly=y,silky=k,smooth=s

13. **stalk-surface-below-ring:** fibrous=f,scaly=y,silky=k,smooth=s

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

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

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

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

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

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

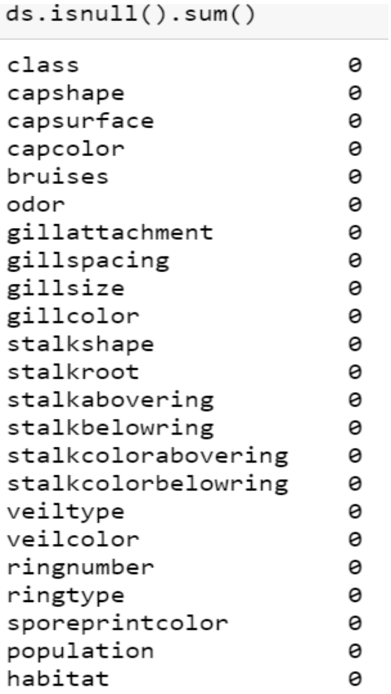
20.**spore-print**-color:black=k,brown=n,buff=b,chocolate=h,green=r,
orange=o,purple=u,white=w,yellow=y

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

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

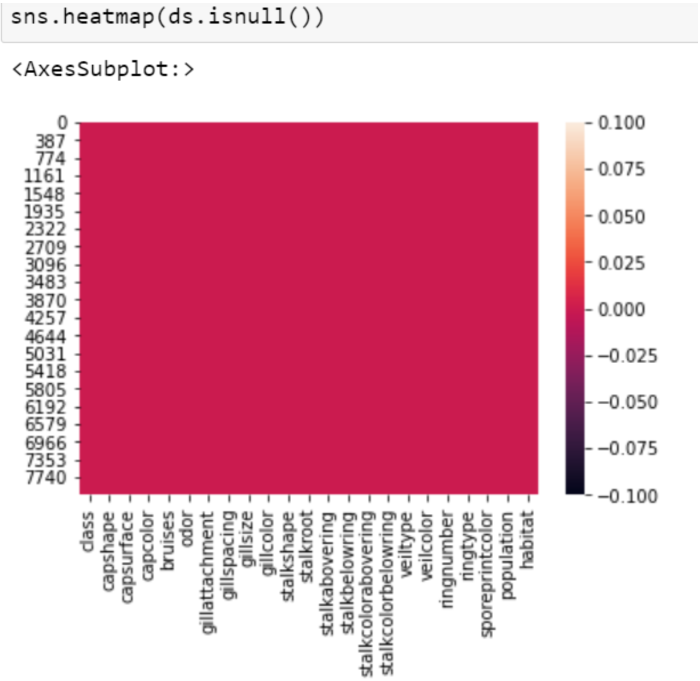
2.3 **Missing Values:** There is no missing values in the data set.

Heat Map



```
ds.isnull().sum()
```

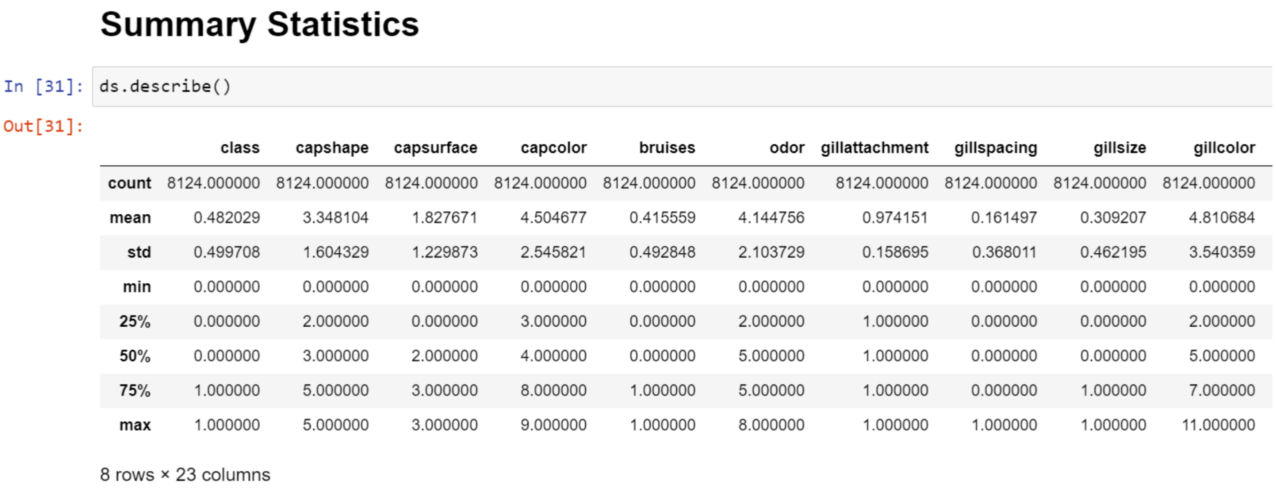
```
class                   0
capshape                0
capsurface              0
capcolor                0
bruises                 0
odor                    0
gillattachment          0
gillspacing             0
gillsize                0
gillcolor               0
stalkshape              0
stalkroot               0
stalkabovering          0
stalkbelowring          0
stalkcolorabovering     0
stalkcolorbelowring     0
veiltype                0
veilcolor               0
ringnumber              0
ringtype                0
sporeprintcolor         0
population              0
habitat                 0
```



```
sns.heatmap(ds.isnull())
```

```
<AxesSubplot:>
```

2.4 Statistical Summary



Summary Statistics

```
In [31]: ds.describe()
```

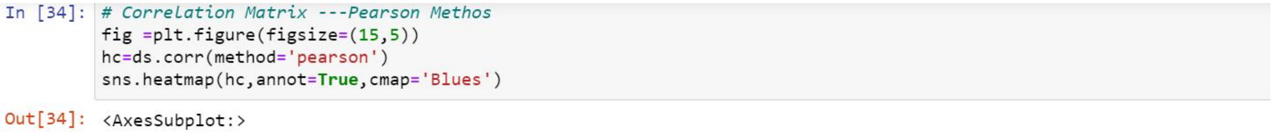
Out[31]:

	class	capshape	capsurface	capcolor	bruises	odor	gillattachment	gillspacing	gillsize	gillcolor
count	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000	8124.000000
mean	0.482029	3.348104	1.827671	4.504677	0.415559	4.144756	0.974151	0.161497	0.309207	4.810684
std	0.499708	1.604329	1.229873	2.545821	0.492848	2.103729	0.158695	0.368011	0.462195	3.540359
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	3.000000	0.000000	2.000000	1.000000	0.000000	0.000000	2.000000
50%	0.000000	3.000000	2.000000	4.000000	0.000000	5.000000	1.000000	0.000000	0.000000	5.000000
75%	1.000000	5.000000	3.000000	8.000000	1.000000	5.000000	1.000000	0.000000	1.000000	7.000000
max	1.000000	5.000000	3.000000	9.000000	1.000000	8.000000	1.000000	1.000000	1.000000	11.000000

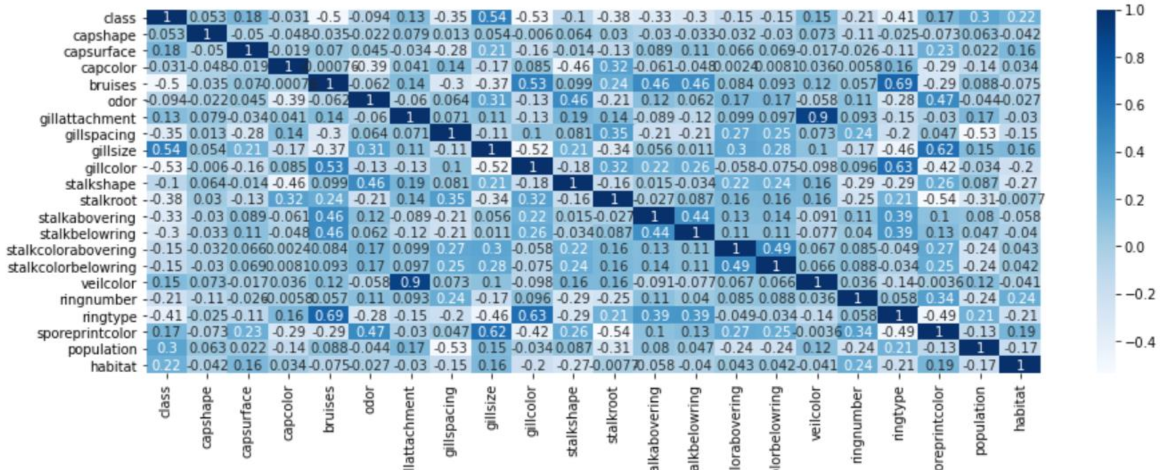
8 rows × 23 columns

Key Observations:1) From the table above it can be seen that the column "veiltype" is not contributing to the data, so we can drop it. 2) The mean and median (50%) having similar values for almost all the variables. And also it can be observe that there is not much difference between the values of 75% and max for the above variables. It helps us to predict that the data is normalized and outliers are absent from the data set.

2.5 Correlation Matrix ---Pearson Method:

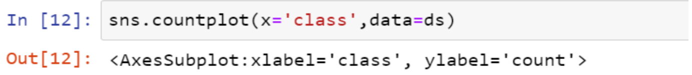


```
In [34]: # Correlation Matrix ---Pearson Methos
         fig =plt.figure(figsize=(15,5))
         hc=ds.corr(method='pearson')
         sns.heatmap(hc,annot=True,cmap='Blues')

Out[34]: <AxesSubplot:>
```

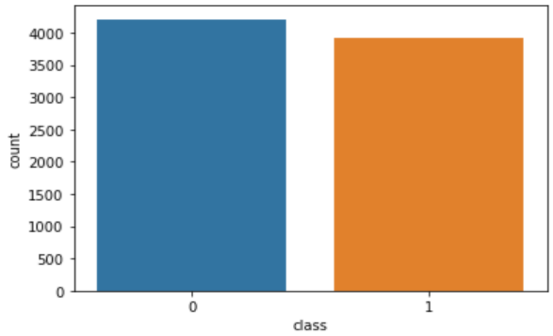


Key Observations: 1)*velicolur* and *gillattachment* variables are highly positively correlated 2)*gillsize* is positively correlated with class variable 3) *bruises, gilltype* variables are positively correlated with *ringtype* variable.

2.6 Checking Class Imbalance: There is no class imbalance in the target variable.

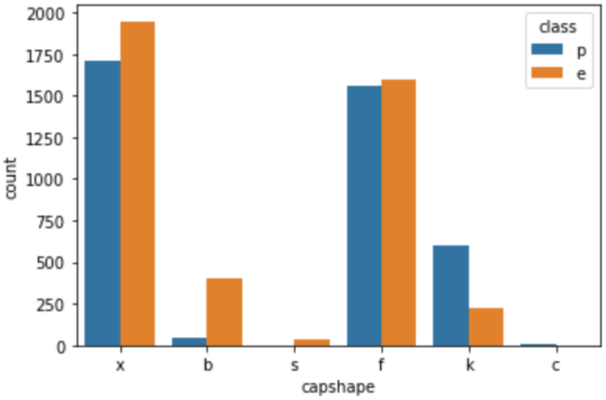


```
In [12]: sns.countplot(x='class',data=ds)

Out[12]: <AxesSubplot:xlabel='class', ylabel='count'>
```

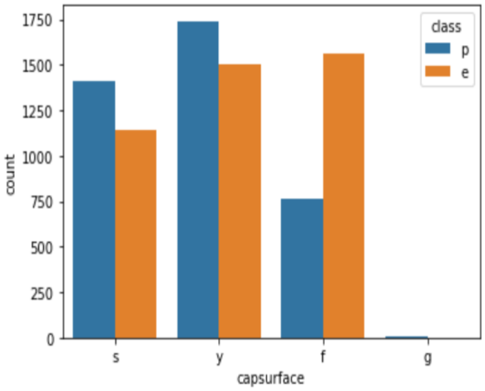


2.7 Checking count “Target variable vs Other variables”

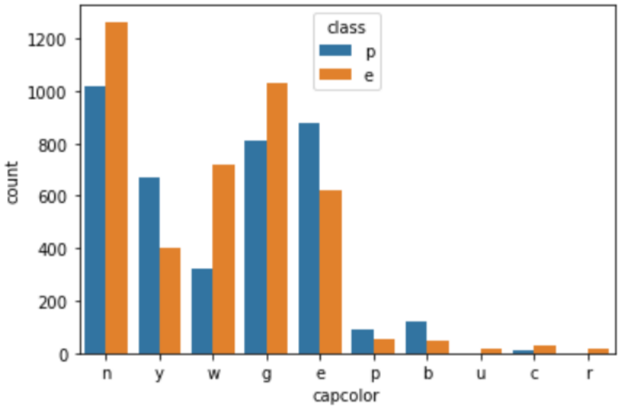
```
<AxesSubplot:xlabel='capshape', ylabel='count'>
```



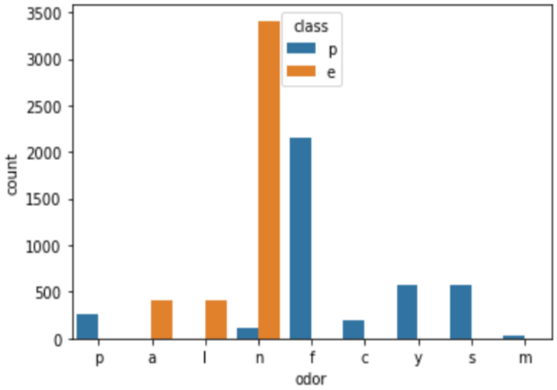
```
<AxesSubplot:xlabel='capsurface', ylabel='count'>
```



```
<AxesSubplot:xlabel='capcolor', ylabel='count'>
```



```
<AxesSubplot:xlabel='odor', ylabel='count'>
```

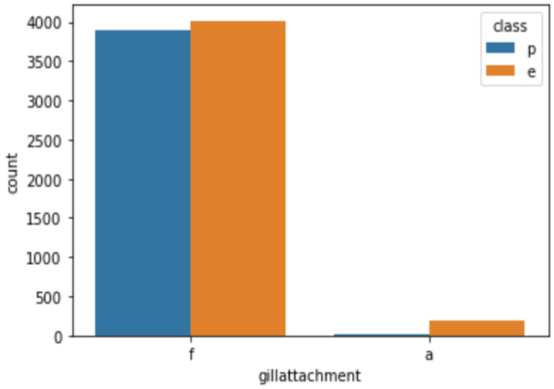


Key Observations:1) Convex & flat shape mushrooms are equally poisonous and edible.

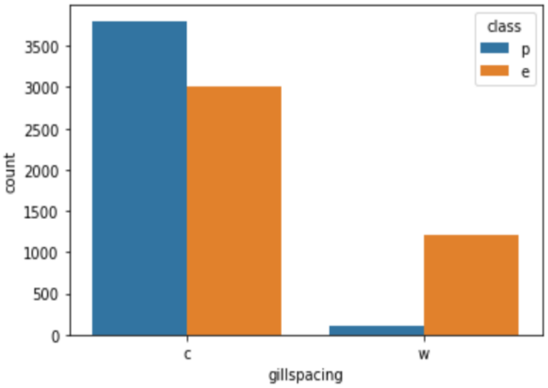
2) Most of mushrooms in Red, White & Purple color are poisonous in nature.

3) Fishy, Foul and Spicy odor mushrooms are poisonous in nature.

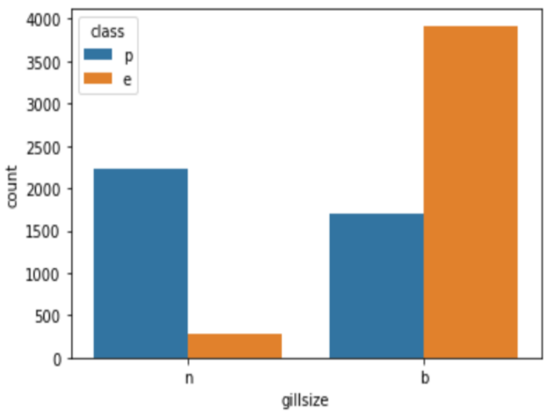
```
<AxesSubplot:xlabel='gillattachment', ylabel='count'>
```



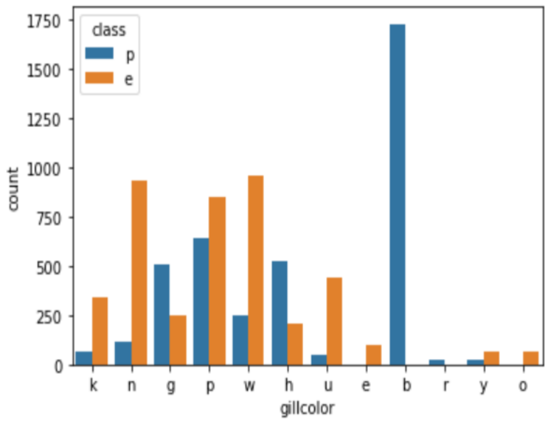
```
<AxesSubplot:xlabel='gillspacing', ylabel='count'>
```



```
<AxesSubplot:xlabel='gillsize', ylabel='count'>
```



```
<AxesSubplot:xlabel='gillcolor', ylabel='count'>
```



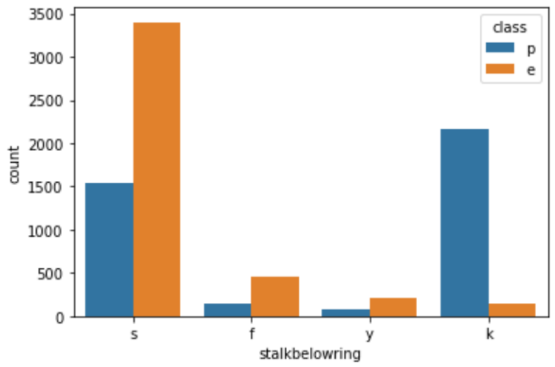
Key Observations1): gill free mushrooms are equally poisonous and edible

2) The close space gill mushrooms are poisonous in nature.

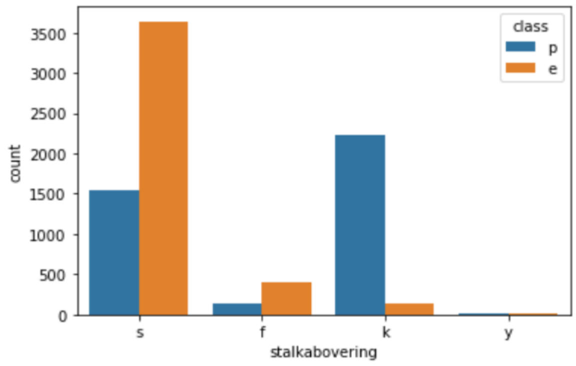
3) Most of the narrow gill size mushrooms are poisonous in nature.

4) Chocolate & buff color mushrooms are poisonous in nature.

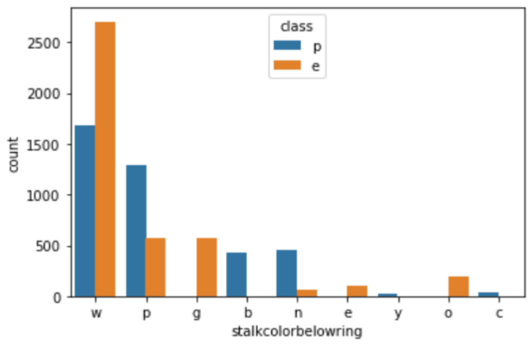
```
<AxesSubplot:xlabel='stalkbelowring', ylabel='count'>
```



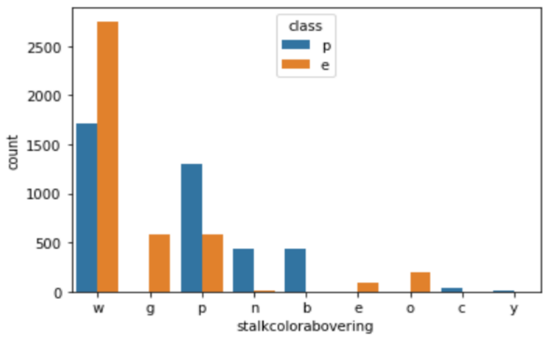
```
<AxesSubplot:xlabel='stalkabovering', ylabel='count'>
```



```
<AxesSubplot:xlabel='stalkcolorbelowring', ylabel='count'
```

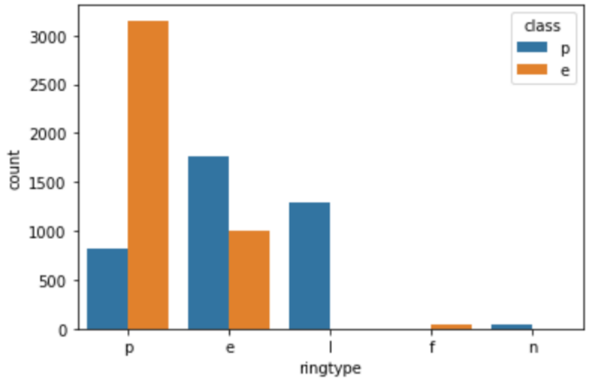


```
<AxesSubplot:xlabel='stalkcolorabovering', ylabel='count'
```

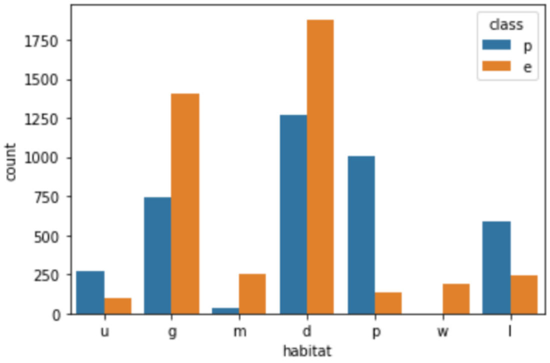


Key Observations:1) The most of stalk *surface* below& above rink ***silky*** mushrooms are poisonous. 2) The most of *stalk color* below& above rink ***pink*** mushrooms are poisonous.

```
<AxesSubplot:xlabel='ringtype', ylabel='count'>
```



```
<AxesSubplot:xlabel='habitat', ylabel='count'>
```



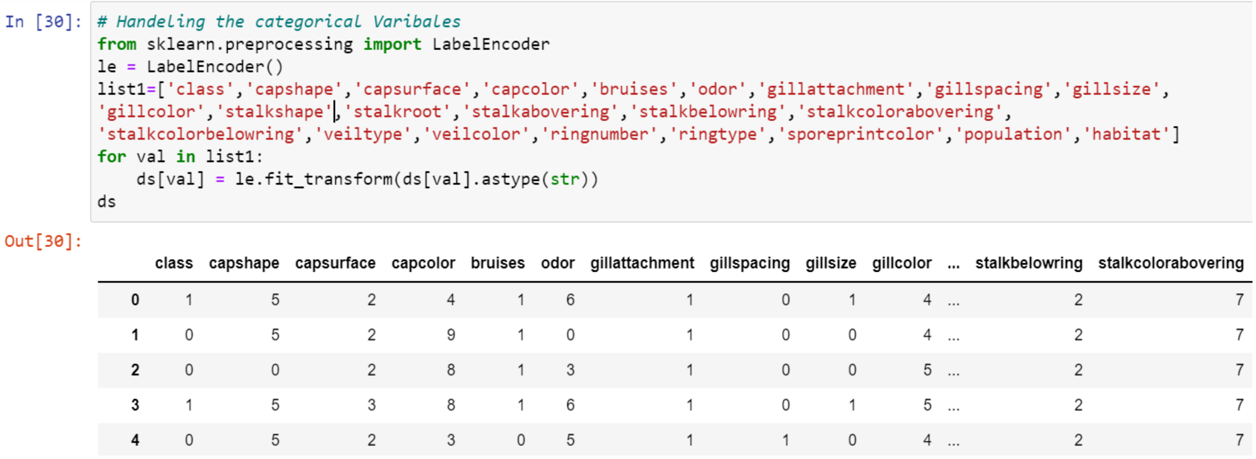
Key Observations:1) The most of the ***mushrooms*** with habitat at path or woods are poisonous.

2) Most of the Large& evanescent rink type mushrooms are poisonous in nature

EDA Summary: The above graphs show that the features like gill, stalk and cap and their attributes have high correlation with the class of the mushrooms. Secondly, the habitat, ring types and odor have significant relationship with target variable ‘class’ of mushroom.

3 Data Preprocessing :

3.1 Handling the categorical Variables: Since we have all the variables categorical, we need to encode them first using label encoding method.

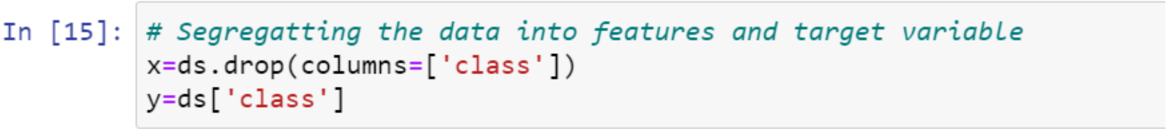


```
In [30]: # Handeling the categorical Varibales
         from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         list1=['class','capshape','capsurface','capcolor','bruises','odor','gillattachment','gillspacing','gillsize',
         'gillcolor','stalkshape','stalkroot','stalkabovering','stalkbelowring','stalkcolorabovering',
         'stalkcolorbelowring','veiltype','veilcolor','ringnumber','ringtype','sporeprintcolor','population','habitat']
         for val in list1:
             ds[val] = le.fit_transform(ds[val].astype(str))
         ds
```

Out[30]:

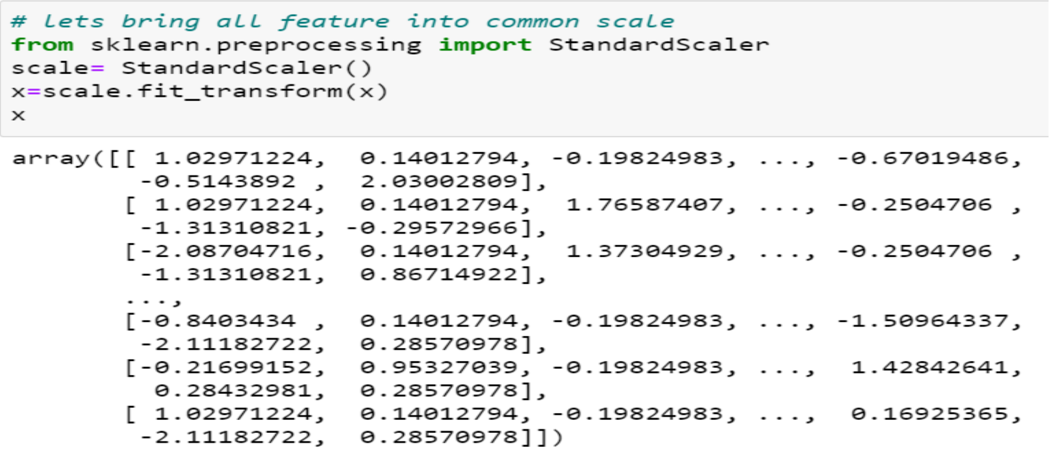
	class	capshape	capsurface	capcolor	bruises	odor	gillattachment	gillspacing	gillsize	gillcolor	...	stalkbelowring	stalkcolorabovering
0	1	5	2	4	1	6	1	0	1	4	...	2	7
1	0	5	2	9	1	0	1	0	0	4	...	2	7
2	0	0	2	8	1	3	1	0	0	5	...	2	7
3	1	5	3	8	1	6	1	0	1	5	...	2	7
4	0	5	2	3	0	5	1	1	0	4	...	2	7

4 Final Dataframe: After preprocessing, we now split the data into training/testing subsets.



```
In [15]: # Segregatting the data into features and target variable
         x=ds.drop(columns=['class'])
         y=ds['class']
```

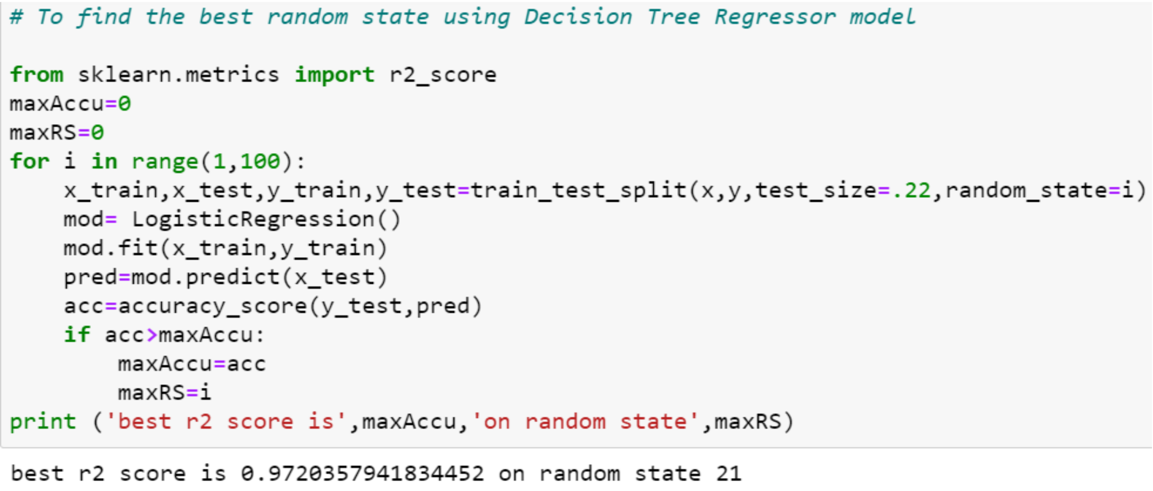
4.1 Scaling dataset: using standard scalar transformation.



```
# Lets bring all feature into common scale
from sklearn.preprocessing import StandardScaler
scale= StandardScaler()
x=scale.fit_transform(x)
x
```

```
array([[ 1.02971224,  0.14012794, -0.19824983, ..., -0.67019486,
        -0.5143892 ,  2.03002809],
       [ 1.02971224,  0.14012794,  1.76587407, ..., -0.2504706 ,
        -1.31310821, -0.29572966],
       [-2.08704716,  0.14012794,  1.37304929, ..., -0.2504706 ,
        -1.31310821,  0.86714922],
       ...,
       [-0.8403434 ,  0.14012794, -0.19824983, ..., -1.50964337,
        -2.11182722,  0.28570978],
       [-0.21699152,  0.95327039, -0.19824983, ...,  1.42842641,
         0.28432981,  0.28570978],
       [ 1.02971224,  0.14012794, -0.19824983, ...,  0.16925365,
        -2.11182722,  0.28570978]])
```

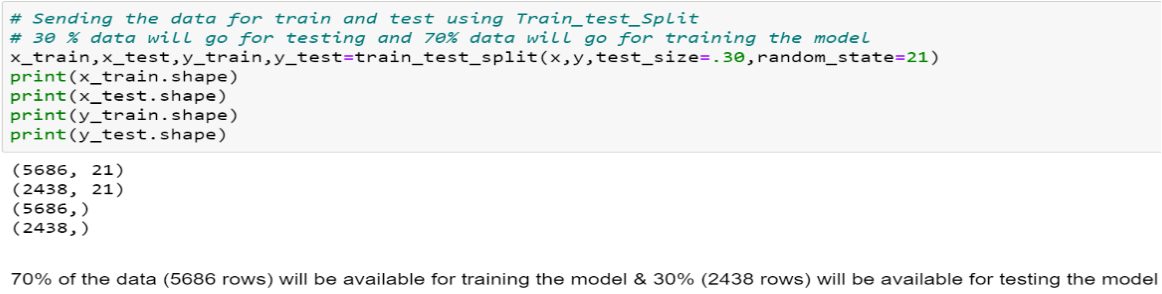
4.2 To find the best random state using ***DecisionTree regressor***



```
# To find the best random state using Decision Tree Regressor model

from sklearn.metrics import r2_score
maxAccu=0
maxRS=0
for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.22,random_state=i)
    mod= LogisticRegression()
    mod.fit(x_train,y_train)
    pred=mod.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print ('best r2 score is',maxAccu,'on random state',maxRS)
```

```
best r2 score is 0.9720357941834452 on random state 21
```

4.3 Train Test Split: split the data into training/testing subsets.



```
# Sending the data for train and test using Train_test_Split
# 30 % data will go for testing and 70% data will go for training the model
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=21)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

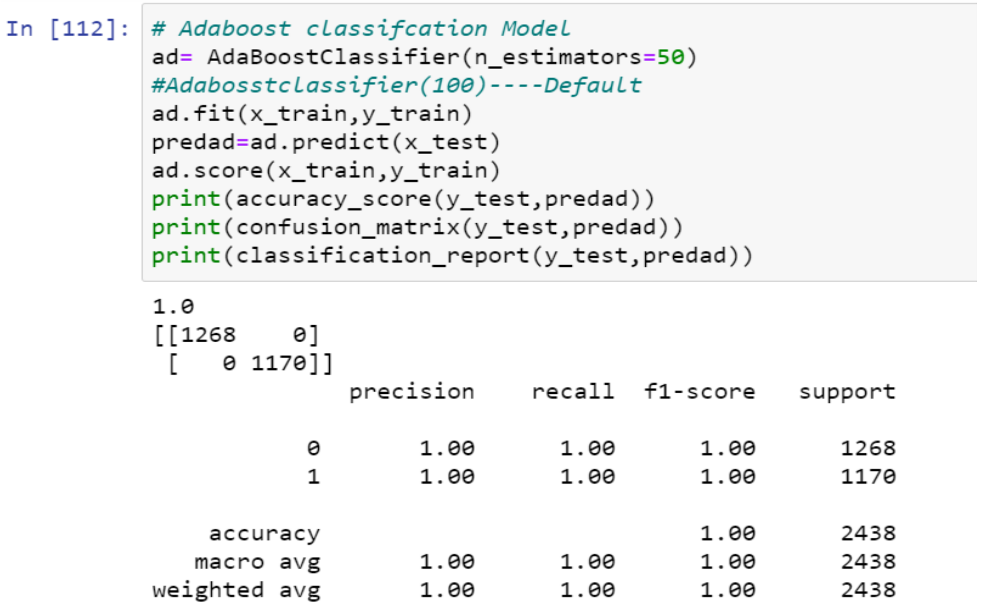
```
(5686, 21)
(2438, 21)
(5686,)
(2438,)
```

70% of the data (5686 rows) will be available for training the model & 30% (2438 rows) will be available for testing the model

5 Model Building & Evaluation

Since the target variable is categorical. we can build the classification models. Therefore our evaluation Matrices will be accuracy_score, confusion_matrix, classifcation_report, AUC-ROC curve.

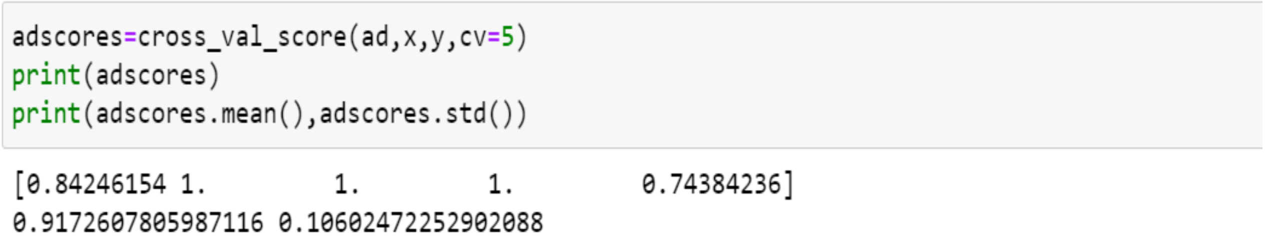
We will be adopting AdaBoost Classification model.



```
In [112]: # Adaboost classifcation Model
          ad= AdaBoostClassifier(n_estimators=50)
          #Adabosstclassifier(100)----Default
          ad.fit(x_train,y_train)
          predad=ad.predict(x_test)
          ad.score(x_train,y_train)
          print(accuracy_score(y_test,predad))
          print(confusion_matrix(y_test,predad))
          print(classification_report(y_test,predad))

          1.0
          [[1268    0]
           [   0 1170]]
                        precision    recall  f1-score   support

                     0       1.00      1.00      1.00      1268
                     1       1.00      1.00      1.00      1170

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

The AdaBoost model gives 100% accuracy and perfect f1 and precision scores.

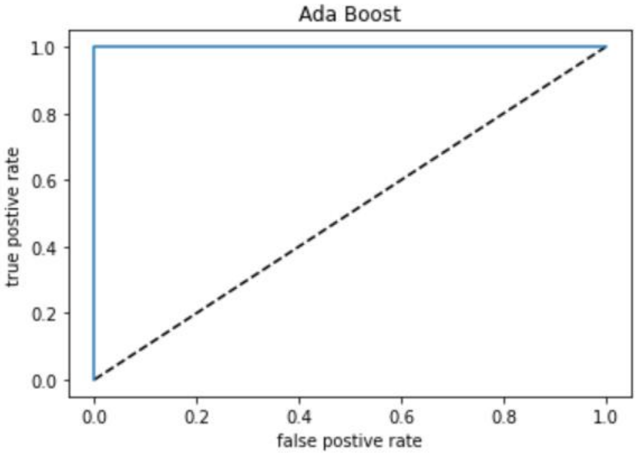
5.1 **Cross Validation:**



```
adscores=cross_val_score(ad,x,y,cv=5)
print(adscores)
print(adscores.mean(),adscores.std())

[0.84246154 1.         1.         1.         0.74384236]
0.9172607805987116 0.10602472252902088
```

It can be seen from the above that the accuracy score of the AdaBoost classifier model is similar before and after the cross validation.

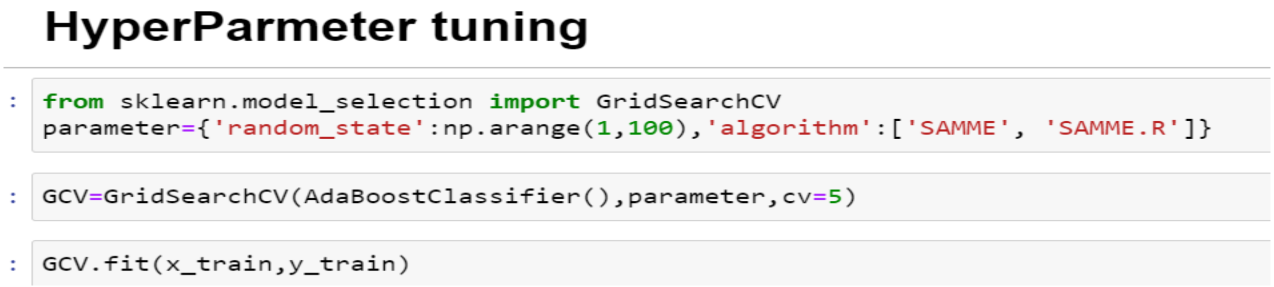
5.2 **AUC-ROC Curve:**



```
Out[114]: 1.0
```

We can depict from the above figure that the are under the AUC-ROC curve is 1 for the model which means the model is 100% efficient for detecting the toxicity of the mushrooms.

5.3

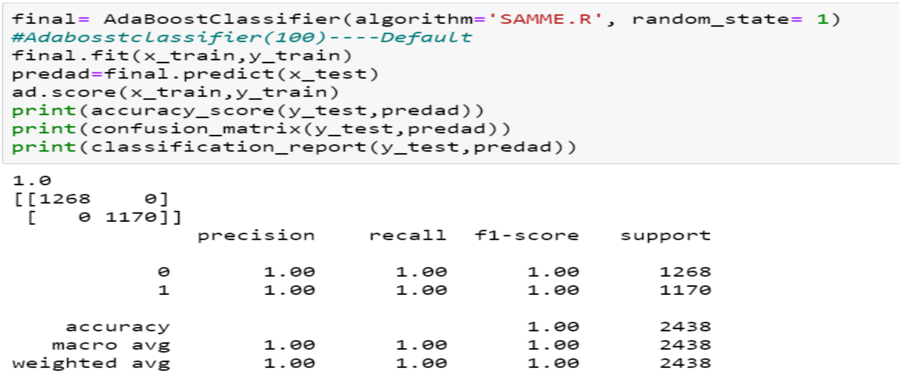


HyperParmeter tuning

```
from sklearn.model_selection import GridSearchCV
parameter={'random_state':np.arange(1,100),'algorithm':['SAMME', 'SAMME.R']}
```

```
GCV=GridSearchCV(AdaBoostClassifier(),parameter,cv=5)
```

```
GCV.fit(x_train,y_train)
```



```
final= AdaBoostClassifier(algorithm='SAMME.R', random_state= 1)
#Adabosstclassifier(100)----Default
final.fit(x_train,y_train)
predad=final.predict(x_test)
ad.score(x_train,y_train)
print(accuracy_score(y_test,predad))
print(confusion_matrix(y_test,predad))
print(classification_report(y_test,predad))
```

```
1.0
[[1268    0]
 [   0 1170]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1268
           1       1.00      1.00      1.00      1170

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

This is the final model for the deployment purpose.

6 Conclusion

We have trained a good model with perfect score for predicting the toxicity of mushrooms. Firstly, we tried to understand the key attributes that helps in the better classification of mushrooms by EDA and preprocessing of the data. Some of the features like color, odor, ring type, habitat, gill, stalk and cap have contributed the most for training the model with good accuracy. The AdaBoost Classifier model has given the perfect accuracy score 1 and perfect area under AUC-ROC curve 1. Our future work shall focus on extracting more physical dimension form the mushroom and try to expand the database to improve the classification process.