In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/mushroom-attributes/mushroom.csv
```

Loading CSV file onto dataframe

In [2]:

```
df=pd.read_csv('/kaggle/input/mushroom-attributes/mushroom.csv')
```

In [3]:

```
def transform(df,x):
    dict={}
    count=0
    for i in (df[x].unique()):
        dict[i]=count
        count+=1
    for i in range(0,len(df[x].unique())+1):
        dict[i]=i
    for i in df[x]:
        df.loc[df[x] == i, x] = dict[i]
```

In [4]:

```
for i in df:
    transform(df,i)
```

In [5]:

```
list(df['cap-shape'].unique())
```

Out[5]:

```
[0, 1, 2, 3, 4, 5]
```

Converting datatype of Columns of dataframe into integer

In [6]:

```
for i in df:
    uniq = df[i].unique()
    new = []
    for j in df[i]:
        new.append(np.where(uniq==j)[0][0])

    df[i] = new
    df.head()
```

In [7]:

```
df.head()
```

Out[7]:

	cap-shape	cap-surface	cap-color	bruises%3F	odor	gill-attachment	gill-spacing	gill-size	gill-color	stalk-shape	...	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-numbe
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	0	0	1	0	1	0	0	1	0	0	...	0	0	0	0	0
2	1	0	2	0	2	0	0	1	1	0	...	0	0	0	0	0
3	0	1	2	0	0	0	0	0	1	0	...	0	0	0	0	0
4	0	0	3	1	3	0	1	1	0	1	...	0	0	0	0	0

5 rows × 23 columns

In [8]:

```
y = df['class']
X =df.drop(['class'], axis=1)
```

In [9]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.2,random_state=2003)
```

Creating Pipeline along with model training using different classification algos

In [10]:

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

In [11]:

```
pipelines = {
    'lr':make_pipeline(StandardScaler(), LogisticRegression()),
    'rc':make_pipeline(StandardScaler(), RidgeClassifier()),
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier()),
    'gnb':make_pipeline(StandardScaler(), GaussianNB()),
    'dtc':make_pipeline(StandardScaler(),DecisionTreeClassifier()),
    'xg':make_pipeline(StandardScaler(),XGBClassifier()),
    'svc':make_pipeline(StandardScaler(),SVC())
}
```

In [12]:

```
fit_models = {}
```

```
for algo, pipeline in pipelines.items():
    model = pipeline.fit(X_train, y_train)
    fit_models[algo] = model
```

Evaluating Performance of various models on test dataset

In [13]:

```
from sklearn.metrics import accuracy_score
al=[]
ac=[]
for algo, model in fit_models.items():
    yhat = model.predict(X_test)
    al.append(algo)
    ac.append(accuracy_score(y_test, yhat))
    print(algo, accuracy_score(y_test, yhat))
```

```
lr 0.9907692307692307
rc 0.9581538461538461
rf 1.0
gb 1.0
gnb 0.9163076923076923
dtc 1.0
xg 1.0
svc 1.0
```

In [14]:

```
for i in range(0,len(ac)):
      ac[i]=round(ac[i]*100,2)
ac
```

Out[14]:

```
[99.08, 95.82, 100.0, 100.0, 91.63, 100.0, 100.0, 100.0]
```

In [15]:

```
al
```

Out[15]:

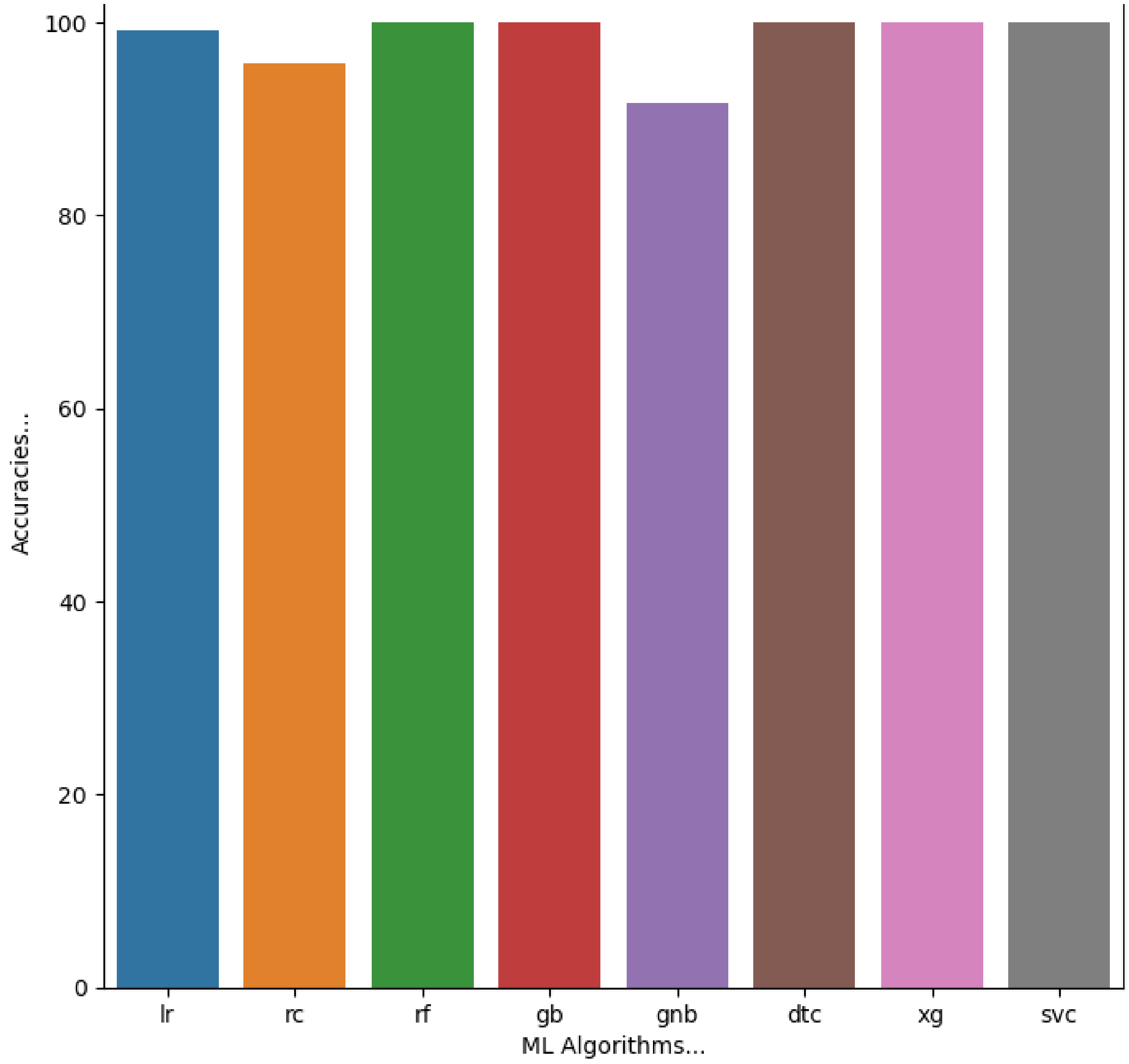
```
['lr', 'rc', 'rf', 'gb', 'gnb', 'dtc', 'xg', 'svc']
```

In [16]:

```
def plot_sns(raw_data,xdata):
  data = np.array(raw_data)

  x = np.array(xdata)
  width = 0.2  # width of bar

  sns.axes_style('white')
  sns.set_style('white')

  ax = sns.barplot(x, data[:,0])
```

In [17]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

plt.figure(figsize=(8,8))
plt.xlabel('ML Algorithms...')
plt.ylabel('Accuracies...')
ax=sns.barplot(x=al,y=ac)
plt.show()
```



Cross Validation

In [18]:

```
from sklearn.model_selection import cross_val_score
import warnings
warnings.filterwarnings("ignore")
for i in al:
    score_lr=cross_val_score(pipelines[i][1],X_train, y_train,cv=20)
    print("Avg :",np.average(score_lr),i)
```

```
Avg : 0.9815360873694206 lr
Avg : 0.9613793922127256 rc
Avg : 1.0 rf
Avg : 1.0 gb
Avg : 0.9293722697056029 gnb
Avg : 1.0 dtc
Avg : 1.0 xg
Avg : 0.9989226020892689 svc
```

Do upvote the notebook if you like the work