In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df=pd.read_csv("iris.data.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   5.1          149 non-null    float64
 1   3.5          149 non-null    float64
 2   1.4          149 non-null    float64
 3   0.2          149 non-null    float64
 4   Iris-setosa  149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	5.1	3.5	1.4	0.2
count	149.000000	149.000000	149.000000	149.000000
mean	5.848322	3.051007	3.774497	1.205369
std	0.828594	0.433499	1.759651	0.761292
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [6]:

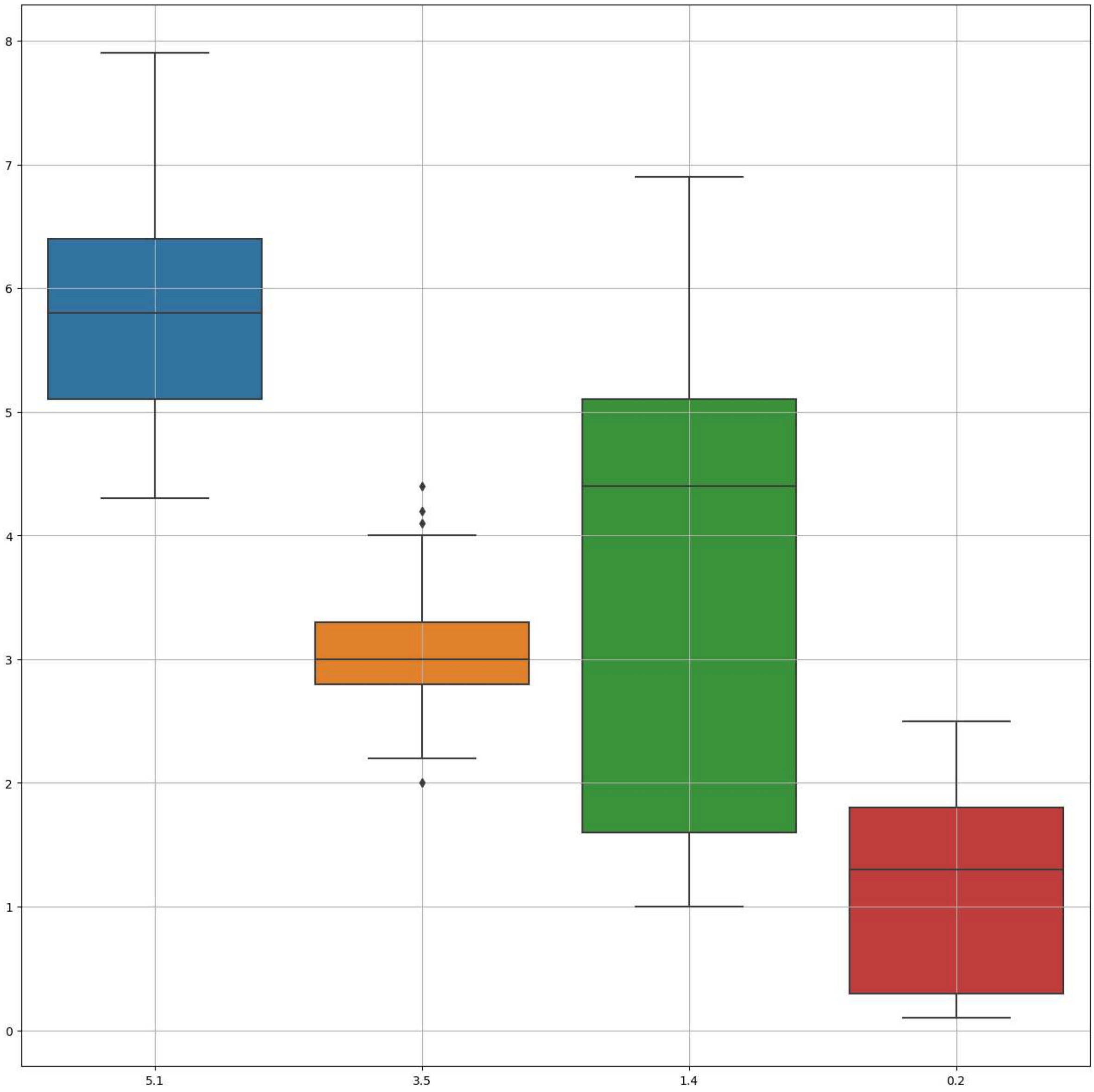
```
df.isna().sum()
```

Out[6]:

```
5.1            0
3.5            0
1.4            0
0.2            0
Iris-setosa    0
dtype: int64
```

In [7]:

```
plt.figure(figsize=(16,16))
sns.boxplot(data=df)
plt.grid()
```



In [8]:

```
df[df["3.5"]>4]
```

Out[8]:

	5.1	3.5	1.4	0.2	Iris-setosa
14	5.7	4.4	1.5	0.4	Iris-setosa
31	5.2	4.1	1.5	0.1	Iris-setosa
32	5.5	4.2	1.4	0.2	Iris-setosa

In [9]:

```
df.drop([14,31,32],axis=0,inplace=True)
```

In [10]:

```
df[df["3.5"]==2]
```

Out[10]:

	5.1	3.5	1.4	0.2	Iris-setosa
59	5.0	2.0	3.5	1.0	Iris-versicolor

In [11]:

```
df.drop(59,axis=0,inplace=True)
```

In [12]:

```
df.head()
```

Out[12]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

In [13]:

```
features=df.iloc[:,:-1]
```

In [14]:

```
features
```

Out[14]:

	5.1	3.5	1.4	0.2
0	4.9	3.0	1.4	0.2
1	4.7	3.2	1.3	0.2
2	4.6	3.1	1.5	0.2
3	5.0	3.6	1.4	0.2
4	5.4	3.9	1.7	0.4
...	...	...	...	...
144	6.7	3.0	5.2	2.3
145	6.3	2.5	5.0	1.9
146	6.5	3.0	5.2	2.0
147	6.2	3.4	5.4	2.3
148	5.9	3.0	5.1	1.8

145 rows × 4 columns

In [15]:

```
target=df.iloc[:,-1]
```

In [16]:

```
target
```

Out[16]:

```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
144    Iris-virginica
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
Name: Iris-setosa, Length: 145, dtype: object
```

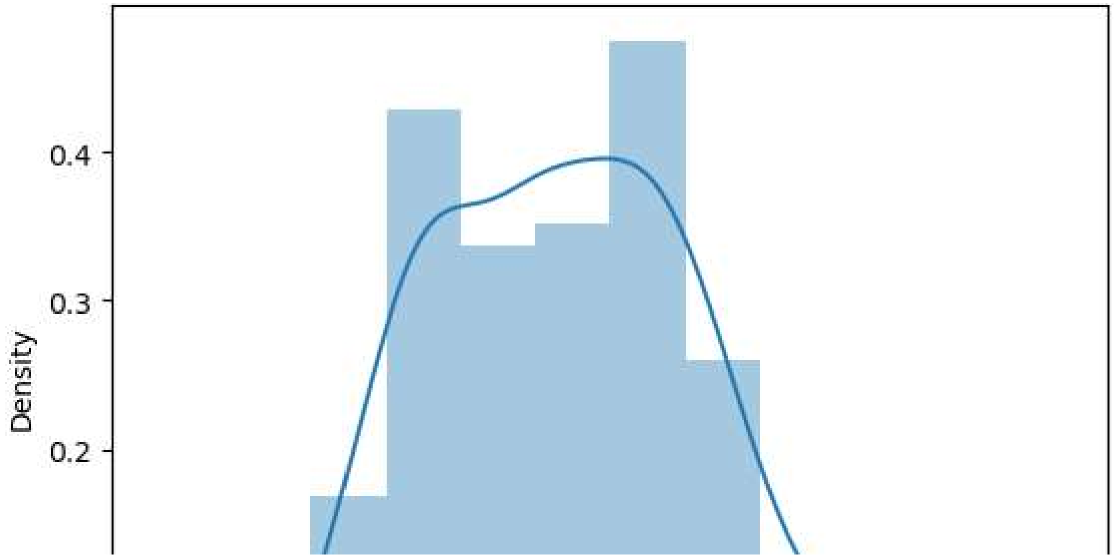
In [17]:

```
from scipy.stats import skew
```

In [18]:

```
for i in features:
    print(i)
    print(skew(features[i]))
    plt.figure()
    sns.distplot(features[i])
    plt.show()
```

```
5.1
0.2633445967124602
```



In [19]:

```
features["5.1"]=np.log(features["5.1"])
```

In [20]:

```
features["3.5"]=np.log(features["3.5"])
```

In [21]:

```
from sklearn.preprocessing import LabelEncoder
```

In [22]:

```
one=LabelEncoder()
```

In [23]:

```
target=one.fit_transform(target)
```

In [24]:

```
target
```

Out[24]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [25]:

```
from sklearn.preprocessing import StandardScaler
```

In [26]:

```
sd=StandardScaler()
```

In [27]:

```
features=sd.fit_transform(features)
```

In [28]:

```
features=pd.DataFrame(features)
features
```

Out[28]:

	0	1	2	3
0	-1.192999	-0.020770	-1.388693	-1.359047
1	-1.486786	0.472581	-1.445979	-1.359047
2	-1.638402	0.229885	-1.331407	-1.359047
3	-1.050572	1.372948	-1.388693	-1.359047
4	-0.508006	1.984817	-1.216834	-1.094357
...	...	...	...	...
140	1.012713	-0.020770	0.788177	1.420200
141	0.578737	-1.414488	0.673605	0.890820
142	0.799064	-0.020770	0.788177	1.023165
143	0.465937	0.936013	0.902749	1.420200
144	0.116285	-0.020770	0.730891	0.758474

145 rows × 4 columns

In [29]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [ ]:

```

```

In [30]:

```
from sklearn.model_selection import train_test_split
```

In [31]:

```
xtrain,xtest,ytrain,ytest=train_test_split(features,target,test_size=0.3,random_state=1)
```

In [32]:

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
```

In [33]:

```
from sklearn.metrics import classification_report
```

In [34]:

```
def pro(project):
    project.fit(xtrain,ytrain)
    ypred=project.predict(xtest)

    train=project.score(xtrain,ytrain)
    test=project.score(xtest,ytest)

    print(f"Training Acc:{train}\n Testing Acc:{test}\n\n")
    print(classification_report(ytest,ypred))
    return project
```

In [35]:

```
gnb=pro(GaussianNB())
```

```
Training Acc:0.9504950495049505
 Testing Acc:1.0


              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        44
   macro avg       1.00      1.00      1.00        44
weighted avg       1.00      1.00      1.00        44
```

In [36]:

```
bnb=pro(BernoulliNB())
```

```
Training Acc:0.7425742574257426
 Testing Acc:0.7727272727272727


              precision    recall  f1-score   support

           0       0.86      1.00      0.92        18
           1       0.71      0.38      0.50        13
           2       0.69      0.85      0.76        13

    accuracy                           0.77        44
   macro avg       0.75      0.74      0.73        44
weighted avg       0.76      0.77      0.75        44
```

In [37]:

```
from sklearn.svm import SVC
```

In [38]:

```
svm=SVC()
svm.fit(xtrain,ytrain)
ypred=svm.predict(xtest)
```

In [39]:

```
svm=pro(SVC())
```

```
Training Acc:0.9504950495049505
 Testing Acc:1.0


              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        44
   macro avg       1.00      1.00      1.00        44
weighted avg       1.00      1.00      1.00        44
```

In [40]:

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        44
   macro avg       1.00      1.00      1.00        44
weighted avg       1.00      1.00      1.00        44
```

In [41]:

```
train=svm.score(xtrain,ytrain)
test=svm.score(xtest,ytest)
```

In [ ]:

In [43]:

```
svc = SVC()
```

In [44]:

```
from sklearn.model_selection import GridSearchCV
```

In [45]:

```
parameter={
    "C":[0.1,1,10],
    "gamma":[0.1,0.01,0.001],
    "kernel":["rbf"]
}
```

In [46]:

```
grid=GridSearchCV(SVC(),parameter,verbose=3)
```

In [47]:

```
grid.fit(xtrain,ytrain)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.905 total time=
0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.850 total time=
0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.800 total time=
0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.800 total time=
0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.750 total time=
0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.619 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.700 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.350 total time=
0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.350 total time=
0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.350 total time=
0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.619 total time=
0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.700 total time=
0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.350 total time=
0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.350 total time=
0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.350 total time=
0.0s
[CV 1/5] END .......C=1, gamma=0.1, kernel=rbf;, score=1.000 total time=
0.0s
[CV 2/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.950 total time=
0.0s
[CV 3/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.900 total time=
0.0s
[CV 4/5] END .......C=1, gamma=0.1, kernel=rbf;, score=1.000 total time=
0.0s
[CV 5/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.800 total time=
0.0s
[CV 1/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.857 total time=
0.0s
[CV 2/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.850 total time=
0.0s
[CV 3/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.850 total time=
0.0s
[CV 4/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.950 total time=
0.0s
[CV 5/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.700 total time=
0.0s
[CV 1/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.619 total time=
0.0s
[CV 2/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.700 total time=
0.0s
[CV 3/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.350 total time=
0.0s
```

```
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.350 total time=
0.0s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.350 total time=
0.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=
0.0s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.950 total time=
0.0s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.900 total time=
0.0s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=
0.0s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.950 total time=
0.0s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total time=
0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.850 total time=
0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.950 total time=
0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=1.000 total time=
0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.800 total time=
0.0s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.905 total time=
0.0s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.900 total time=
0.0s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.850 total time=
0.0s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.950 total time=
0.0s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.700 total time=
0.0s
```

Out[47]:

```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10], 'gamma': [0.1, 0.01, 0.001],
                         'kernel': ['rbf']},
             verbose=3)
```

In [48]:

```
grid.best_params_
```

Out[48]:

```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

In [49]:

```
grid.best_score_
```

Out[49]:

```
0.96
```

In [50]:

```
grid.best_estimator_
```

Out[50]:

```
SVC(C=10, gamma=0.1)
```

In [51]:

```
svm=grid.best_estimator_
svm.fit(xtrain,ytrain)
ypred=svm.predict(xtest)
```

In [52]:

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        44
   macro avg       1.00      1.00      1.00        44
weighted avg       1.00      1.00      1.00        44
```

In [53]:

```
train=svm.score(xtrain,ytrain)
test=svm.score(xtest,ytest)
```

In [54]:

```
print(f"train acc:{train}\n test acc:{test}")
```

```
train acc:0.9702970297029703
 test acc:1.0
```

In [ ]:

```

```