ANKIT GAUTAM

TASK -1 IRIS FLOWER CLASSIFICATION

Iris flower has three species; setosa, versicolor, and virginica, which differs according to their measurements. Now assume that you have the measurements of the iris flowers according totheir species, and here your task is to train a machine learning model that can learn from themeasurements of the iris species and classify them.

In [1]:

```
import pandas as pd #importing lib
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
df = pd.read_csv('iris.csv')
```

In [3]:

```
df.tail()
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [4]:

```
df.shape
```

Out[4]:

```
(150, 5)
```

In [5]:

```
df['species'].value_counts()
```

Out[5]:

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

In [6]:

```
df.describe()
```

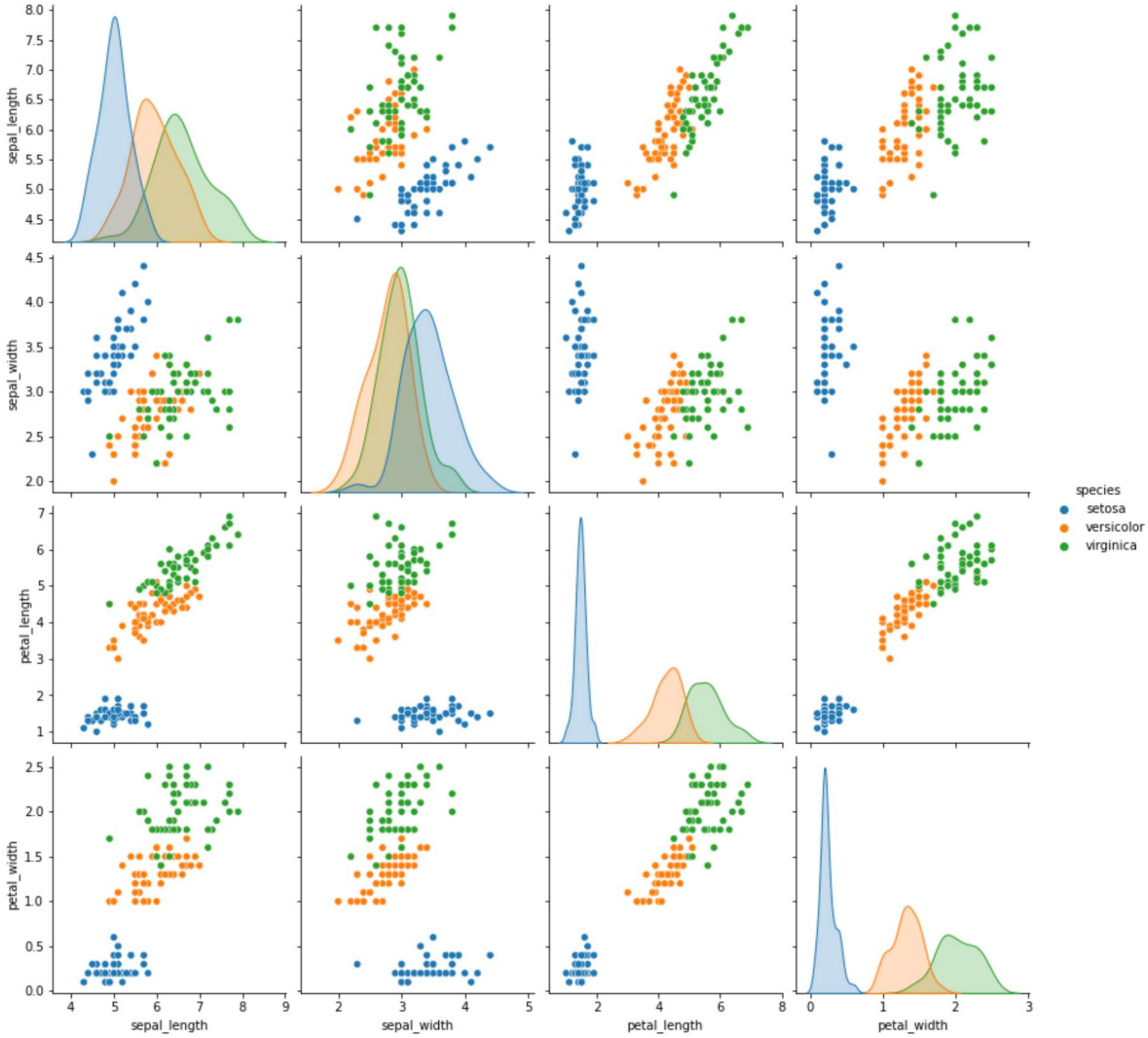
Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

To understand how each feature accounts for classification of the data, we can build a pair-plot which shows us the correlation with respect to other features.

In [7]:

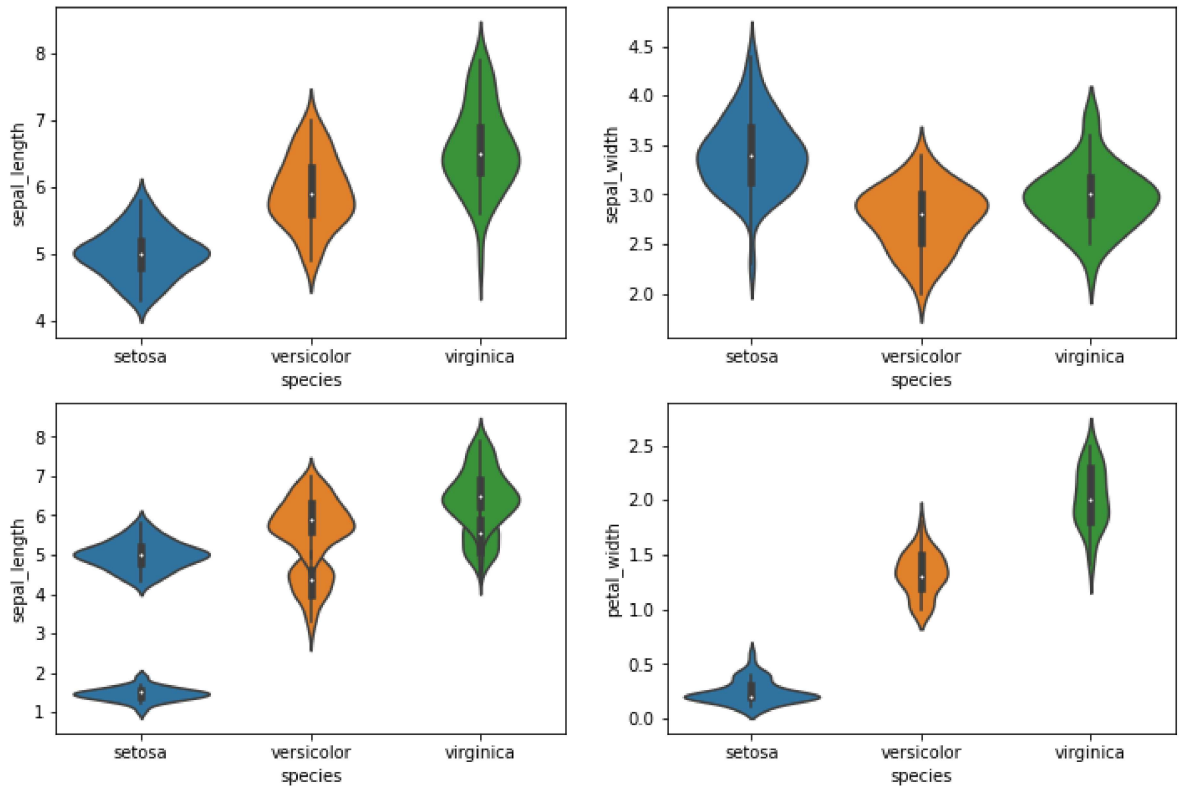
```
sns.pairplot(df,hue= 'species', height=3,aspect=1)
plt.show()
```



we can see that all the attributes are plotted against each other and the three different colours shows the distribution of three individual species (setosa, versicolor and verginica). It shows the distinctive relationships between the attribute

In [8]:

```
plt.figure(figsize=(12,8))
plt.subplot(2,2,1)
sns.violinplot(x= "species", y= "sepal_length", data=df)
plt.subplot(2,2,2)
sns.violinplot(x= "species", y= "sepal_width", data=df)
plt.subplot(2,2,3)
sns.violinplot(x= "species", y= "petal_length", data=df)
sns.violinplot(x= "species", y= "sepal_length", data=df)
plt.subplot(2,2,4)
sns.violinplot(x= "species", y= "petal_width", data=df)
plt.show()
```



Violin plot shows us the visual representation of how our data is scattered over the plane. Here we can conclude from the above picture that in sepal length we can see this the distribution in setosa is much smaller than the versicolor and verginica. In sepal width we can examine that the distribution of setosa is 13 widest and also the longest sepal width and longest petal length in comparisons to the other attributes

In [9]:

```
X = df[['sepal_length','sepal_width','petal_length','petal_width']]  # Identify X,y
y = df['species']
```

In [10]:

```
from sklearn.model_selection import train_test_split #spliting the dataset into train & tes
```

In [11]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20)
```

In [12]:

```
X_train.shape
```

Out[12]:

```
(120, 4)
```

In [13]:

```
X_test.shape
```

Out[13]:

```
(30, 4)
```

In [14]:

```
y_train.shape
```

Out[14]:

```
(120,)
```

In [15]:

```
y_test.shape
```

Out[15]:

```
(30,)
```

In [16]:

```
from sklearn.preprocessing import MinMaxScaler  #scaling
```

In [17]:

```
scaler = MinMaxScaler()
```

In [18]:

```
scaler.fit(X_train)
```

Out[18]:

```
MinMaxScaler()
```

In [19]:

```
X_train_scaled = scaler.transform(X_train)
```

In [20]:

```
X_train_scaled
```

Out[20]:

```
array([[0.58333333, 0.375     , 0.57142857, 0.5       ],
       [0.69444444, 0.41666667, 0.78571429, 0.83333333],
       [0.38888889, 1.        , 0.07142857, 0.125     ],
       [0.        , 0.41666667, 0.        , 0.        ],
       [0.58333333, 0.33333333, 0.80357143, 0.83333333],
       [0.55555556, 0.375     , 0.80357143, 0.70833333],
       [0.55555556, 0.125     , 0.58928571, 0.5       ],
       [0.66666667, 0.45833333, 0.58928571, 0.54166667],
       [0.58333333, 0.45833333, 0.78571429, 0.70833333],
       [0.47222222, 0.375     , 0.60714286, 0.58333333],
       [0.08333333, 0.58333333, 0.05357143, 0.08333333],
       [0.38888889, 0.33333333, 0.60714286, 0.5       ],
       [0.19444444, 0.66666667, 0.05357143, 0.04166667],
       [0.16666667, 0.45833333, 0.07142857, 0.        ],
       [0.41666667, 0.29166667, 0.71428571, 0.75      ],
       [0.5       , 0.25      , 0.80357143, 0.54166667],
       [0.22222222, 0.20833333, 0.33928571, 0.41666667],
       [0.13888889, 0.41666667, 0.05357143, 0.        ],
```

In [21]:

```
from sklearn.linear_model import LogisticRegression # train the model
```

In [22]:

```
model = LogisticRegression()
```

In [23]:

```
model.fit(X_train,y_train)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:8
14: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sciki
t-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession)
  n_iter_i = _check_optimize_result(
```

Out[23]:

```
LogisticRegression()
```

In [24]:

```
X_test_scaled = scaler.transform(X_test)
```

In [25]:

```
model.score(X_test,y_test)
```

Out[25]:

```
1.0
```

The model gives 99% accuracy on the test set, which is good

In [26]:

```
model.predict([[6.2,3.4,5.4,2.3]])[0]
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but LogisticRegression was fitted with fea
ture names
  warnings.warn(
```

Out[26]:

```
'virginica'
```

In [ ]:

```

```