Task 1 : Iris Flowers Classification

Author : Mitali D Shinde

Level : Beginner

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn import tree
```

In [3]:
```
iris_data = pd.read_csv('Iris3.csv')
#get the dataset
```

In [4]:
```
iris_data.head()
#display first 5 rows
```

Out[4]:

	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 [5]:
```
iris_data.tail()
#display last 5 rows
```

Out[5]:

	5.1	3.5	1.4	0.2	Iris-setosa
144	6.7	3.0	5.2	2.3	Iris-virginica
145	6.3	2.5	5.0	1.9	Iris-virginica
146	6.5	3.0	5.2	2.0	Iris-virginica
147	6.2	3.4	5.4	2.3	Iris-virginica
148	5.9	3.0	5.1	1.8	Iris-virginica

In [6]:
```
iris_data.columns
#display column heads
```

Out[6]:
```
Index(['5.1', '3.5', '1.4', '0.2', 'Iris-setosa'], dtype='object')
```

In [7]:
```
columns = ['sepal_lenght','sepal_width','petal_lenght','petal_width','Species']
#giving labels to the columns
```

In [8]:
```
iris_data.columns = columns
iris_data.head()
```

Out[8]:

	sepal_lenght	sepal_width	petal_lenght	petal_width	Species
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 [9]:
```
iris_data.shape
#gives size of data
```

Out[9]:
```
(149, 5)
```

In [10]:
```
iris_data.info()
```

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

In [11]:
```
iris_data.describe()
#gives statistical inference about the data
```

Out[11]:

	sepal_lenght	sepal_width	petal_lenght	petal_width
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 [12]:
```
iris_data.isnull().sum()
#gives count of null values
```

Out[12]:
```
sepal_lenght    0
sepal_width     0
petal_lenght    0
petal_width     0
Species         0
dtype: int64
```

data visualization

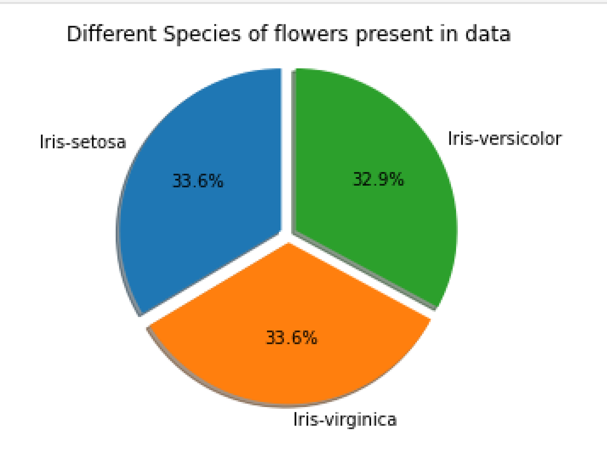
In [13]:
```
count = iris_data['Species'].value_counts()
count.to_frame()
```

Out[13]:

	Species
Iris-versicolor	50
Iris-virginica	50
Iris-setosa	49

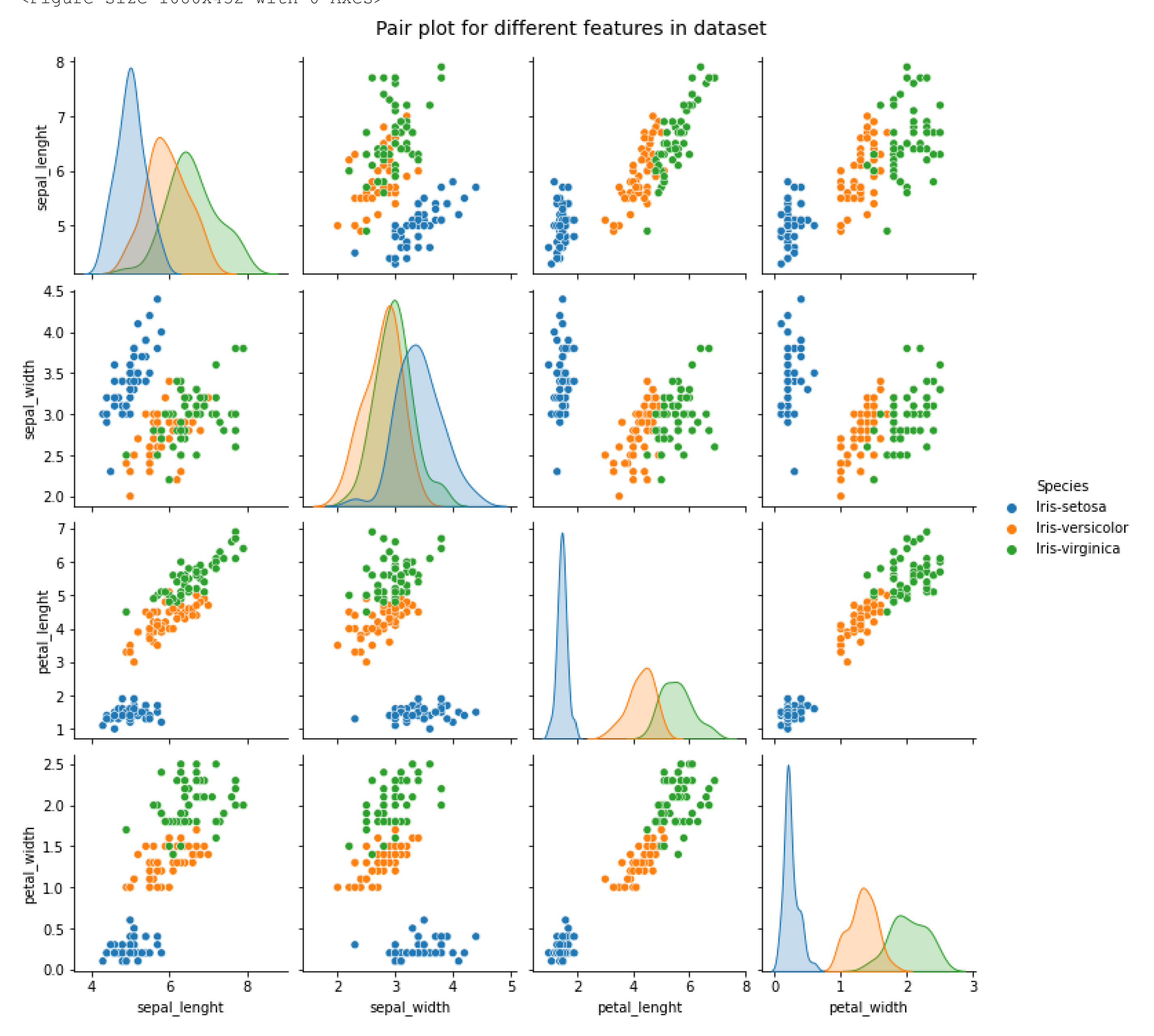
In [14]:
```
labrl =  count.index.tolist()
val=count.values.tolist()
```

In [15]:
```
exp =  (.05, .05,.05)
label = 'Iris-setosa','Iris-virginica','Iris-versicolor'
fig,ax = plt.subplots()
ax.pie(val, explode=exp, labels=label , autopct='%1.1f%%',shadow=True , startangle=90)
plt.title('Different Species of flowers present in data', fontsize=12)
ax.axis('equal')
plt.show()
```

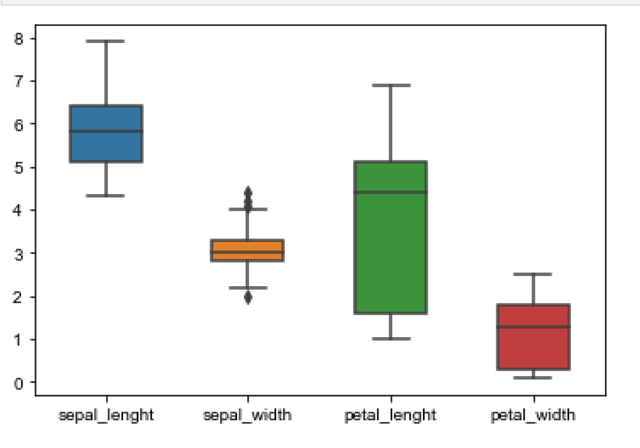


In [16]:
```
fig = plt.figure(figsize=(15,6))
visual =sns.pairplot(iris_data, hue = 'Species')
visual.fig.suptitle("Pair plot for different features in dataset" , y =1.02 ,fontsize =14)
plt.show()
```

```
<Figure size 1080x432 with 0 Axes>
```

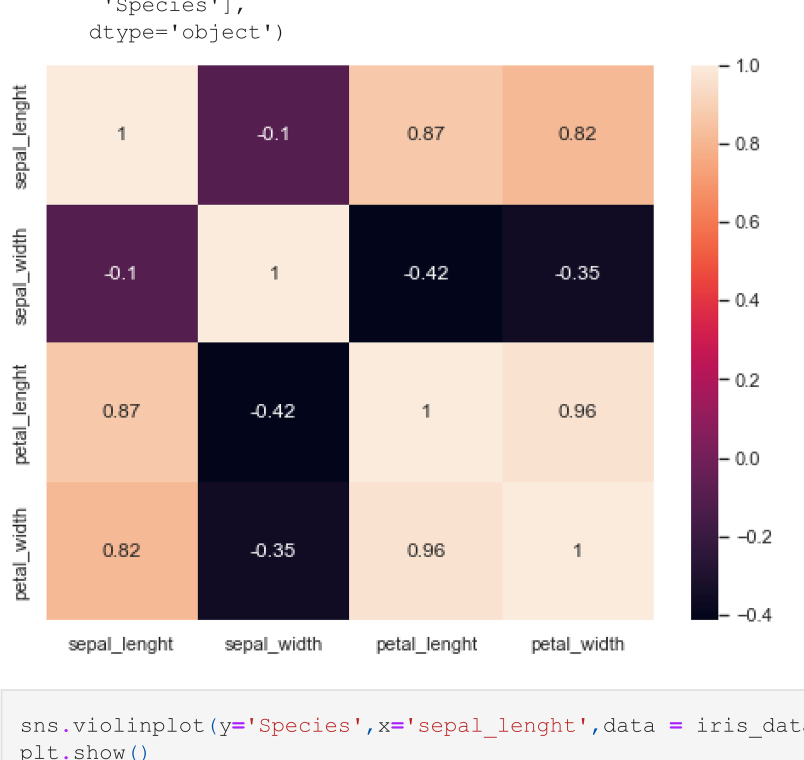


In [17]:
```
sns.boxplot(data = iris_data , width = 0.5 , fliersize = 5)
sns.set(rc = {"figure.figsize" :(6,6)})
```

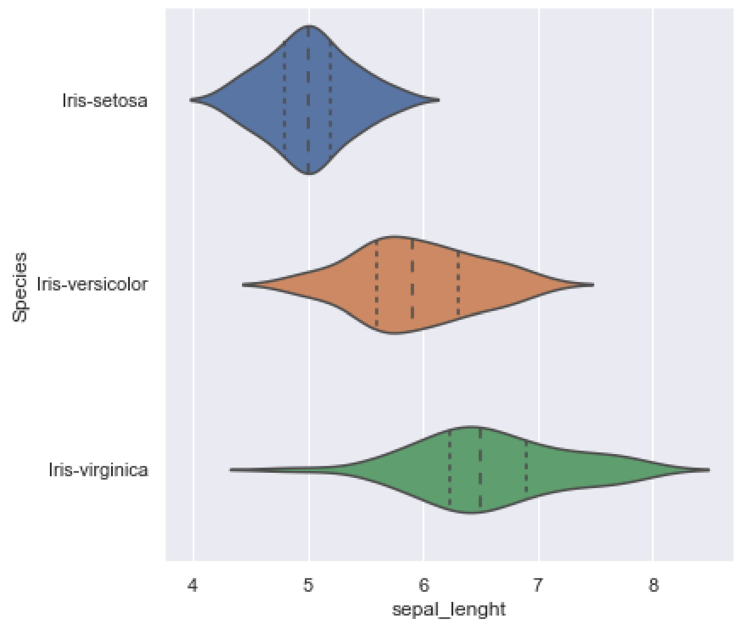


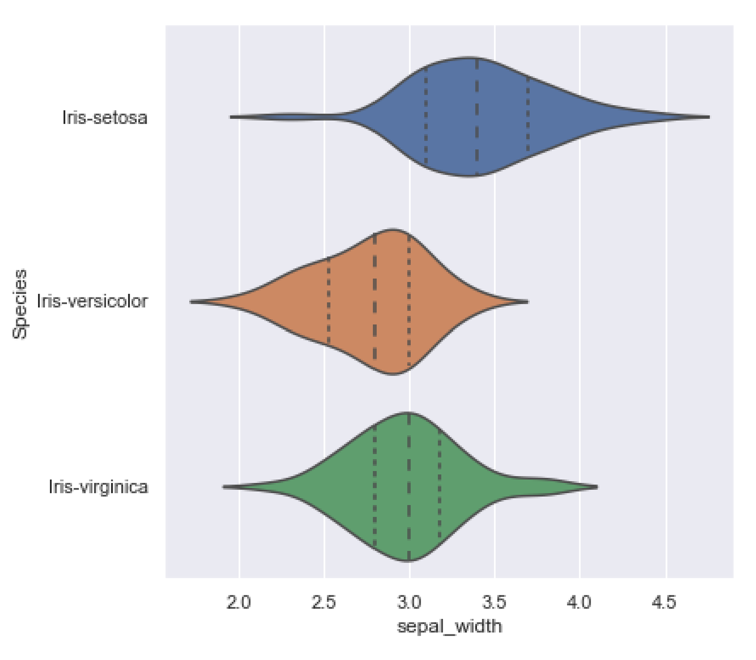
In [18]:
```
corr = iris_data.corr()
plt.figure(figsize=(8,6))
sns.heatmap(corr , annot = True)
iris_data.columns
```

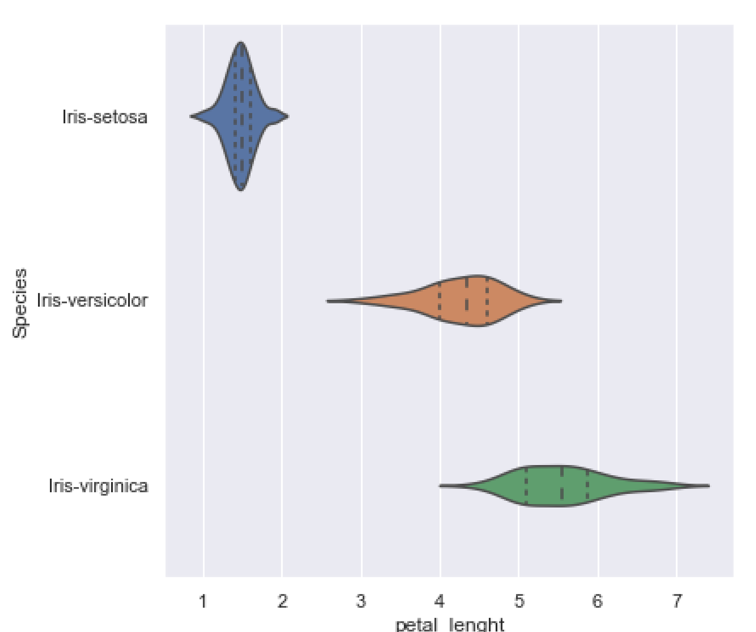
Out[18]:
```
Index(['sepal_lenght', 'sepal_width', 'petal_lenght', 'petal_width',
       'Species'],
      dtype='object')
```

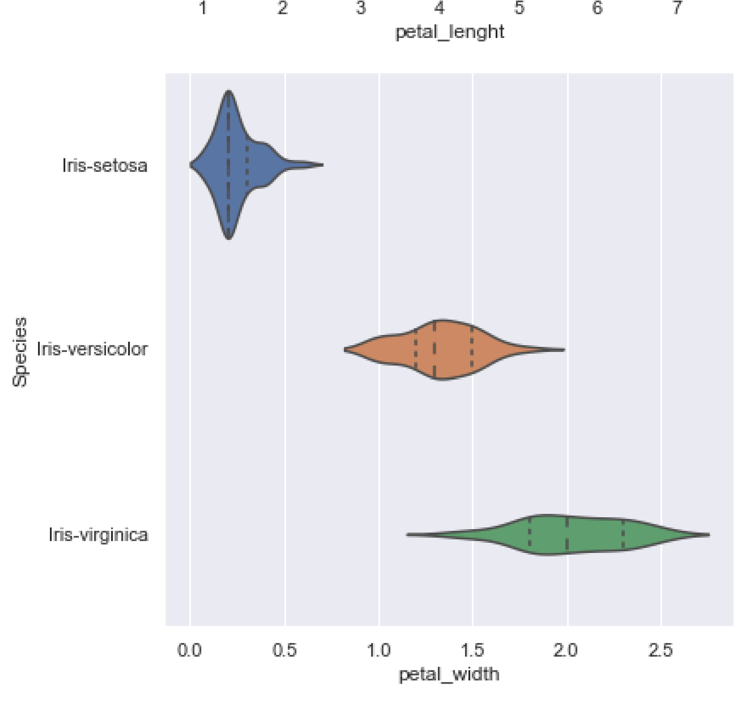


In [19]:
```
sns.violinplot(y='Species',x='sepal_lenght',data = iris_data, inner = 'quartile')
plt.show()
sns.violinplot(y='Species',x='sepal_width',data = iris_data, inner = 'quartile')
plt.show()
sns.violinplot(y='Species',x='petal_lenght',data = iris_data, inner = 'quartile')
plt.show()
sns.violinplot(y='Species',x='petal_width',data = iris_data, inner = 'quartile')
plt.show()
```









In [20]:
```
X = iris_data.drop(['Species'],axis=1)
Y = iris_data['Species']
print(f'X shape : {X.shape} | y shape: {Y.shape}')
```

```
X shape : (149, 4) | y shape: (149,)
```

In [22]:
```
X_train ,X_test , Y_train ,Y_test =  train_test_split(X,Y ,test_size = 0.10 ,random_state =1)
```

Model Acuuracy

In [23]:
```
model =[]
model.append(('SVC',SVC(gamma = 'auto')))
```

In [24]:
```
model = SVC(gamma='auto')
model.fit(X_train,Y_train)
prediction = model.predict(X_test)
```

In [25]:
```
print(f'Test accuracy : {accuracy_score(Y_test , prediction)}')
print(f'Classification Report : \n {classification_report(Y_test , prediction)}')
```

```
Test accuracy : 0.9333333333333333
Classification Report :
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         4
Iris-versicolor       0.90      1.00      0.95         9
 Iris-virginica       1.00      0.50      0.67         2

       accuracy                           0.93        15
      macro avg       0.97      0.83      0.87        15
   weighted avg       0.94      0.93      0.92        15
```

In [ ]: