In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
from warnings import filterwarnings
filterwarnings(action='ignore')
```

In [ ]:
```
df=pd.read_csv("IRIS.csv")
```

In [ ]:
```
df
```

In [ ]:
```
df.info()
```

In [ ]:
```
#check for all null values
df.isna().sum()
```

In [ ]:
```
df.describe()
```

In [ ]:
```
sns.pairplot(df)
```

```
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [ ]:
```
#check for all null values
df.isna().sum()
```

In [ ]:
```
df.describe()
```

In [ ]:
```
sns.pairplot(df)
```

In [ ]:
```
sns.heatmap(df.corr(),annot=True)
```

In [ ]:
```
df.corr()
```

```
petal_width    0
species        0
dtype: int64
```

In [*]:
```
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

In [ ]:
```
sns.pairplot(df)
```

In [ ]:
```
sns.heatmap(df.corr(), annot=True)
```

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

In [*]:
```
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x2bada6a0be0>

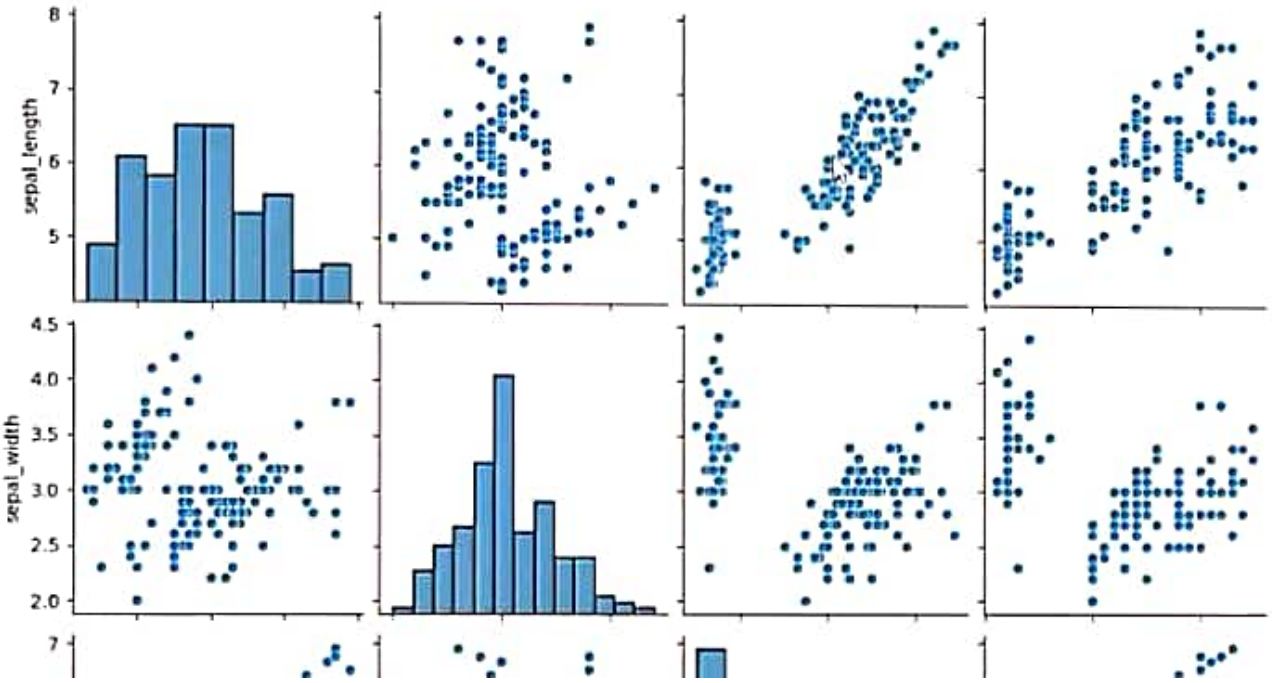
In [ ]:
```
sns.heatmap(df.corr(),annot=True)
```

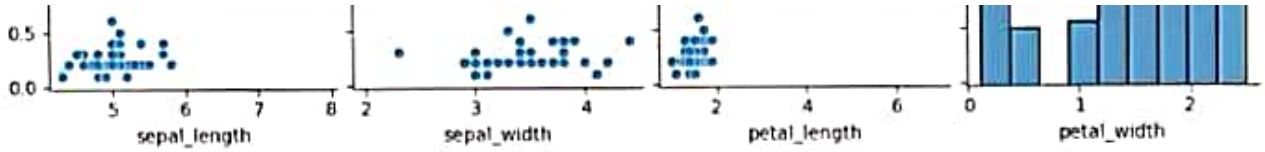
In [ ]:
```
df.corr()
```

In [ ]:
```
#feature selection
x=df.drop(['species'],axis=1)
y=df['species']
```

In [ ]:
```
print(x)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x2bada6a0be0>





In [ ]:
```
sns.heatmap(df.corr(),annot=True)
```

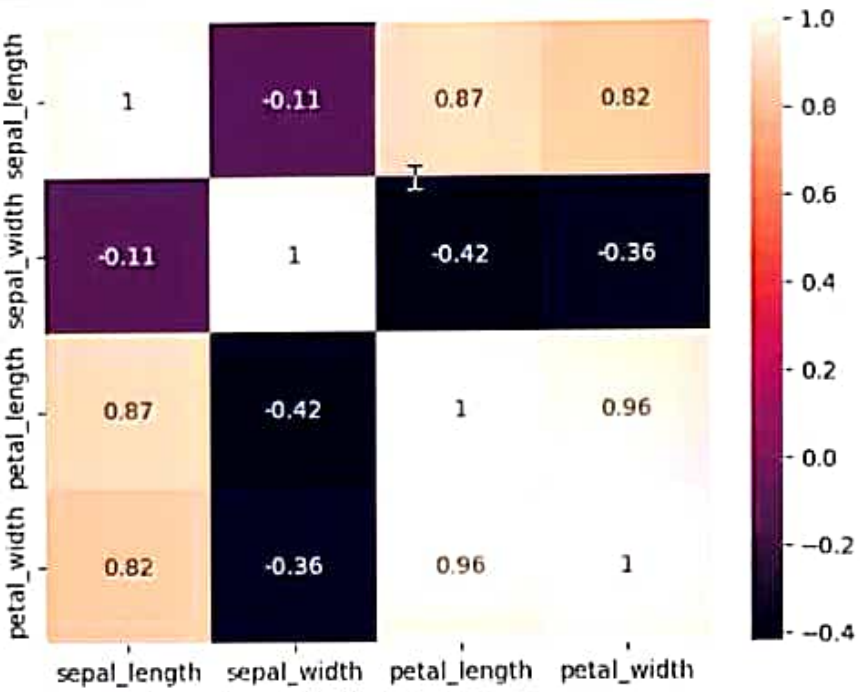
In [ ]:
```
df.corr()
```

In [ ]:
```
#feature selection
x=df.drop(['species'],axis=1)
y=df['species']
```

In [ ]:
```
print(x)
```

In [ ]:
```
print(y)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
 x, y, train_size=0.7, test_size=0.3)
```


sepal_length
sepal_width
petal_length
petal_width
1
-0.11
0.87
0.82
-0.11
1
-0.42
-0.36
0.87
-0.42
1
0.96
0.82
-0.36
0.96
1
sepal_length
sepal_width
petal_length
petal_width
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

In [ ]:
```
df.corr()
```

In [ ]:
```
#feature selection
x=df.drop(['species'],axis=1)
y=df['species']
```

In [ ]:
```
print(x)
```

In [ ]:
```
print(y)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
 x, y, train_size=0.7, test_size=0.3)
```

In [ ]:
```
#Use KNN Classifier
from sklearn.neighbors import KNeighborsClassifier
```

sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

In [ ]:
```
#feature selection
x=df.drop(['species'],axis=1)
y=df['species']
```

In [ ]:
```
print(x)
```

In [ ]:
```
print(y)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
 x, y, train_size=0.7, test_size=0.3)
```

In [ ]:
```
#Use KNN Classifier
from sklearn.neighbors import KNeighborsClassifier
Knn = KNeighborsClassifier(n_neighbors = 3)
```

In [ ]:
```
#Train KNN classifier
```

sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

In [10]:
```
#feature selection
x=df.drop(['species'],axis=1)
y=df['species']
```

In [ ]:
```
print(x)
```

In [ ]:
```
print(y)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
 x, y, train_size=0.7, test_size=0.3)
```

In [ ]:
```
#Use KNN Classifier
from sklearn.neighbors import KNeighborsClassifier
Knn = KNeighborsClassifier(n_neighbors = 3)
```

In [ ]:
```
#Train KNN classifier
```

In [10]:
```
#feature selection
x=df.drop(['species'],axis=1)
y=df['species']
```

In [11]:
```
print(x)
```

```
     sepal_length  sepal_width  petal_length  petal_width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            ...          ...           ...          ...
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
```

In [ ]:
```
print(y)
```

```
149          5.9          3.0          5.1          1.8

[150 rows x 4 columns]
```

In [ ]:
```
print(y)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
 x, y, train_size=0.7, test_size=0.3)
```

In [ ]:
```
#Use KNN Classifier
from sklearn.neighbors import KNeighborsClassifier
Knn = KNeighborsClassifier(n_neighbors = 3)
```

In [ ]:
```
#Train KNN classifier
Knn.fit(X_train,Y_train)
```

In [ ]:
```
#Evaluate Model
Y_pred = Knn.predict(X_test)
```

In [ ]:
```
#check accuracy Score
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,Y_pred)
```

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
 x, y, train_size=0.7, test_size=0.3)
```

In [ ]:
```
#Use KNN Classifier
from sklearn.neighbors import KNeighborsClassifier
Knn = KNeighborsClassifier(n_neighbors = 3)
```

In [ ]:
```
#Train KNN classifier
Knn.fit(X_train,Y_train)
```

In [ ]:
```
#Evaluate Model
Y_pred = Knn.predict(X_test)
```

In [ ]:
```
#check accuracy Score
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,Y_pred)
```

In [ ]: