```
The Project is about predicting the type of flower after training using Machine Learning algorithms(Output will be the species
of Iris Flower)
The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly
separable from the other 2; the latter are NOT linearly separable from each other.
```

```
1 IMPORTING the required libraries
```

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

2 LOADING the dataset: IRIS dataset is easily availble for begginers on internet http://archive.ics.uci.edu/ml/datasets/Iris (http://archive.ics.uci.edu/ml/datasets/Iris)

In [3]:
```
data=sns.load_dataset("iris")
```

Lets explore the dataset little more

In [5]:
```
data.head() # It will give top 5 rows of dataset
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [16]:
```
data.tail() # It will give last 5 rows of dataset
```

Out[16]:

	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 [17]:
```
data.info() #it will give the info about data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
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
```

```
We can observe that

4 columns are sepal_length,sepal_width,petal_length and petal_width seems to be attributes of Dataset

Lets explore colum species
```

In [9]:
```
data['species']
```

Out[9]:
```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...    
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

```
The data shows , there are 150 rows(entries) and species columns shows the type of flower for each kind of class species.In raw
data we observed that there are 3 types of IRIS flower each of 50 instance(iris setosa,iris virginica,iris versicolor)
```

In [18]:
```
data.describe()
```

Out[18]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
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

Dataset shows the statistical values for each of the variables. The data has mean of 5.84 for sepal_length,3.05 for sepal_width,3.75 for petal_length and 1.19 for petal_width.Similarly we can observe min, max ,count and quartile values and standard deviation

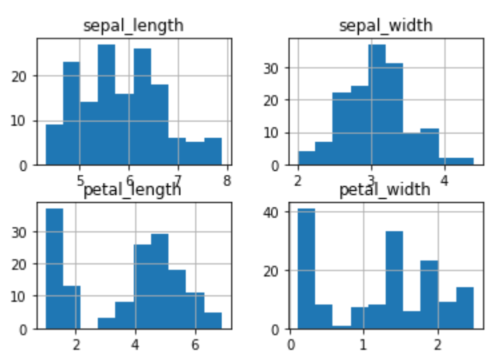
In [20]:
```
data.isnull().sum()*100 # to check null values
```

Out[20]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

There are no null values present in data

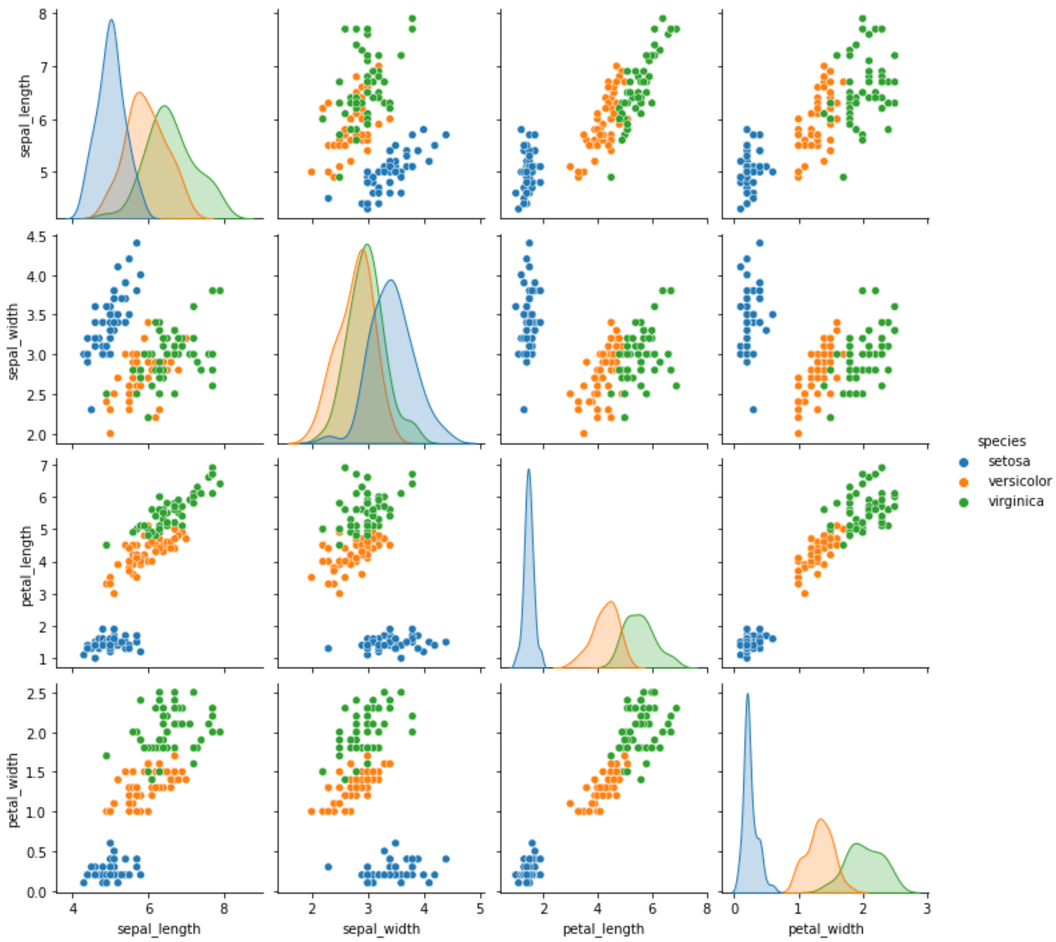
In [21]:
```
data.hist() #to see the disribution of data
```

Out[21]:
```
array([[<AxesSubplot:title={'center':'sepal_length'}>,
        <AxesSubplot:title={'center':'sepal_width'}>],
       [<AxesSubplot:title={'center':'petal_length'}>,
        <AxesSubplot:title={'center':'petal_width'}>]], dtype=object)
```



In [12]:
```
sns.pairplot(data,hue='species')
```

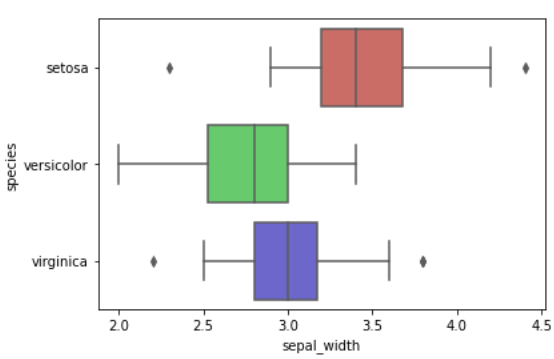
Out[12]: <seaborn.axisgrid.PairGrid at 0x237ee3b7e20>



In [ ]:
```
Paiplot shows the 3 types of flowers relationship in 16 pictures as each of 4 inputs were paired and plotted.

We can see species iris setosa has largest sepal_length,petal_length and petal_width over other species.

Species verginica has largest sepal_width over other species.

So, iris setosa can be visualised more significantly compare to other two.
```

In [37]:
```
sns.boxplot(x='sepal_width',y='species',data=data,palette='hls') #to explore the outliers
```

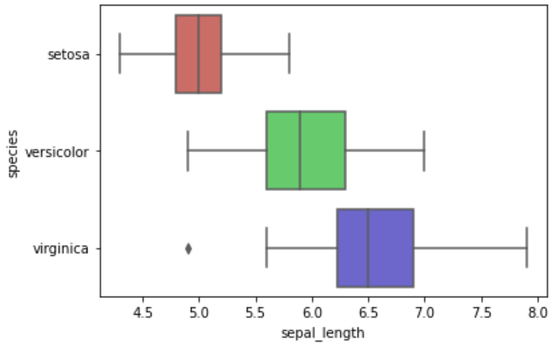
Out[37]: <AxesSubplot:xlabel='sepal_width', ylabel='species'>



In [ ]:
```
we can see outliers are present in setosa and virginica species only.
```

In [38]:
```
sns.boxplot(x='sepal_length',y='species',data=data,palette='hls')
```

Out[38]: <AxesSubplot:xlabel='sepal_length', ylabel='species'>

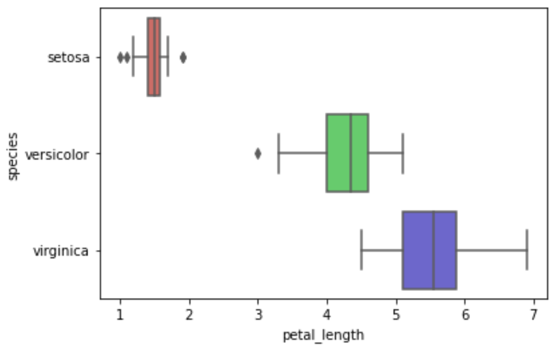


sns.boxplot(x='petal_length',y='species',data=data,palette='hls')

In [ ]:
```
From Boxplot we can observe there are few outliers in virginica
```

In [39]:
```
sns.boxplot(x='petal_length',y='species',data=data,palette='hls')
```

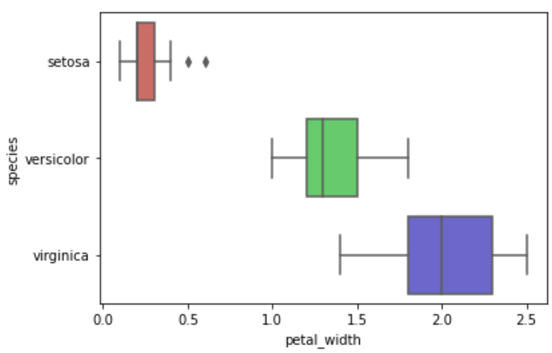
Out[39]: <AxesSubplot:xlabel='petal_length', ylabel='species'>



In [ ]:
```
Outliers found in setosa and vericolor species
```

In [40]:
```
sns.boxplot(x='petal_width',y='species',data=data,palette='hls')
```

Out[40]: <AxesSubplot:xlabel='petal_width', ylabel='species'>



In [ ]:
```
outliers are present in setosa

from over all ouliers we do not find significant outliers.
```

In [13]:
```
#lets perform training and testing by importing libraries
from sklearn.model_selection import train_test_split # It is used to perform training and spliting the test and train data in dat
```

```
Creating matrices for training and testing
```

In [15]:
```
X=data.iloc[: ,:-1] # X is composed of 4 attributes discussed above.
y=data.iloc[:,-1] # y is species (type of flower)
```

In [16]:
```
t,y_train,y_test=train_test_split(X,y,test_size=0.25,stratify=y,random_state=123) # we have taken 25% of dataset for testing by te
```

In [19]:
```
print(X_train)
```

```
     sepal_length  sepal_width  petal_length  petal_width
30            4.8          3.1           1.6          0.2
36            5.5          3.5           1.3          0.2
29            4.7          3.2           1.6          0.2
55            5.7          2.8           4.5          1.3
118           7.7          2.6           6.9          2.3
..            ...          ...           ...          ...
11            4.8          3.4           1.6          0.2
0             5.1          3.5           1.4          0.2
104           6.5          3.0           5.8          2.2
7             5.0          3.4           1.5          0.2
147           6.5          3.0           5.2          2.0

[112 rows x 4 columns]
```

We can see appro. 75% data is taken for training set

In [20]:
```
print(X_test)
```

```
     sepal_length  sepal_width  petal_length  petal_width
135           7.7          3.0           6.1          2.3
34            4.9          3.1           1.5          0.2
61            5.9          3.0           4.2          1.5
117           7.7          3.8           6.7          2.2
42            4.4          3.2           1.3          0.2
38            4.4          3.0           1.3          0.2
65            6.7          3.1           4.4          1.4
125           7.2          3.2           6.0          1.8
80            5.5          2.4           3.8          1.1
19            5.1          3.8           1.5          0.3
64            5.6          2.9           3.6          1.3
33            5.5          4.2           1.4          0.2
115           6.4          3.2           5.3          2.3
146           6.3          2.5           5.0          1.9
94            5.6          2.7           4.2          1.3
116           6.5          3.0           5.5          1.8
28            5.2          3.4           1.4          0.2
32            5.2          4.1           1.5          0.1
9             4.9          3.1           1.5          0.1
17            5.1          3.5           1.4          0.3
40            5.0          3.5           1.3          0.3
22            4.6          3.6           1.0          0.2
93            5.0          2.3           3.3          1.0
144           6.7          3.3           5.7          2.5
2             4.7          3.2           1.3          0.2
77            6.7          3.0           5.0          1.7
122           7.7          2.8           6.7          2.0
138           6.0          3.0           4.8          1.8
110           6.5          3.2           5.1          2.0
56            6.3          3.3           4.7          1.6
66            5.6          3.0           4.5          1.5
101           5.8          2.7           5.1          1.9
68            6.2          2.2           4.5          1.5
76            6.8          2.8           4.8          1.4
105           7.6          3.0           6.6          2.1
86            6.7          3.1           4.7          1.5
127           6.1          3.0           4.9          1.8
92            5.8          2.6           4.0          1.2
```

In [ ]:
```
We observed that, remaining 25% is taken for test set
```

In [43]:
```
from sklearn.preprocessing import MinMaxScaler # for normalizing the dataset on scale
```

In [44]:
```
scaler=MinMaxScaler()# defining the scaler
```

In [45]:
```
X_train=scaler.fit_transform(X_train)
X_test=scaler.fit_transform(X_test)
X_train=pd.DataFrame(X_train,columns=X.columns)
X_test=pd.DataFrame(X_test,columns=X.columns)
# here, I have fit and transform the MinMax scaler on training and testing set.
```

In [46]:
```
X_train.head()
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width
0	0.138889	0.458333	0.086207	0.041667
1	0.333333	0.625000	0.034483	0.041667
2	0.111111	0.500000	0.086207	0.041667
3	0.388889	0.333333	0.586207	0.500000
4	0.944444	0.250000	1.000000	0.916667

In [47]:
```
X_test.head()
```

Out[47]:

	sepal_length	sepal_width	petal_length	petal_width
0	1.000000	0.40	0.894737	0.916667
1	0.151515	0.45	0.087719	0.041667
2	0.454545	0.40	0.561404	0.583333
3	1.000000	0.80	1.000000	0.875000
4	0.000000	0.50	0.052632	0.041667

Model Logistic Regression algorithm

Logistic regression is a classification algorithm. It is used to predict a binary outcome based on a set of independent variables. As we are trying to find type of flower we are using this algorithm.

In [24]:
```
from sklearn.linear_model import LogisticRegression #to import libraries required for model
```

In [25]:
```
lm=LogisticRegression()
lm.fit(X_train,y_train)
y_pred=lm.predict(X_test)
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (st
atus=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://scikit-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-regression (https://scikit-learn.org/stable/modules/line
ar_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [26]:
```
from sklearn.metrics import classification_report,roc_auc_score,confusion_matrix
```

In [27]:
```
print(classification_report(y_pred,y_test),"\n")

print(confusion_matrix(y_pred,y_test))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
  versicolor       0.92      1.00      0.96        12
   virginica       1.00      0.93      0.96        14

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38


[[12  0  0]
 [ 0 12  0]
 [ 0  1 13]]
```

In [14]:
```
print(classification_report(y_train,lm.predict(X_train)))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        38
  versicolor       0.89      0.89      0.89        37
   virginica       0.89      0.89      0.89        37

    accuracy                           0.93       112
   macro avg       0.93      0.93      0.93       112
weighted avg       0.93      0.93      0.93       112
```

Model SVM (Support vector machine)algorithm

SVM is used for gene classification,etc and it can handle linear as well as non linear regressions and classification problems.

In [29]:
```
from sklearn.svm import SVC #to import libraries required for model

svc=SVC()
svc.fit(X_train,y_train)
y_pred=svc.predict(X_test)
```

In [30]:
```
print(classification_report(y_pred,y_test),"\n")

print(confusion_matrix(y_pred,y_test))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
  versicolor       0.92      1.00      0.96        12
   virginica       1.00      0.93      0.96        14

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38


[[12  0  0]
 [ 0 12  0]
 [ 0  1 13]]
```

In [ ]:
```
Model-  DecisionTreeClassifier algorithm
A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks
```

In [40]:
```
from sklearn.tree import DecisionTreeClassifier #to import libraries required for model

dc=DecisionTreeClassifier()
dc.fit(X_train,y_train)
y_pred=dc.predict(X_test)
```

In [41]:
```
print(classification_report(y_pred,y_test),"\n")

print(confusion_matrix(y_pred,y_test))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
  versicolor       0.92      0.86      0.89        14
   virginica       0.85      0.92      0.88        12

    accuracy                           0.92        38
   macro avg       0.92      0.92      0.92        38
weighted avg       0.92      0.92      0.92        38


[[12  0  0]
 [ 0 12  2]
 [ 0  1 11]]
```

In [ ]:
```
Conclusion: From all the above 3 algorithms used, we can find all the models predictions are showing 97% accuracy with
    precision  of 100%,92% and 100% for setoa, versicolor and virginica respectively.
```