Iris Flower Classification using Machine Learning,

iris dataset from Scikit-learn library is loaded using Seaborn

```
import seaborn as sns
```

1. **Data Collection**

```
a=sns.load_dataset('iris')
```

```
a.head(3)
```

	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

```
a.tail(3)
```

	sepal_length	sepal_width	petal_length	petal_width	species
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

Checking for null values

```
a.info() # implies the absence of null values in the dataset
```

```
<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
```

Checking for duplicates

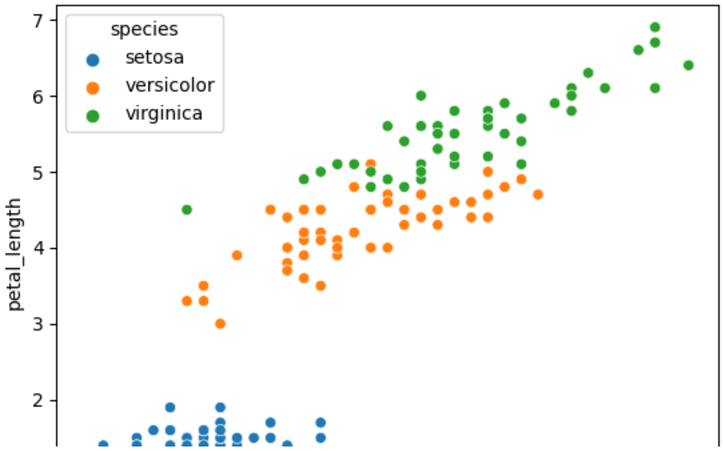
```
a.duplicated() # no duplicates
```

```
0      False
1      False
2      False
3      False
4      False
       ...  
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

2. **Data Visualization**

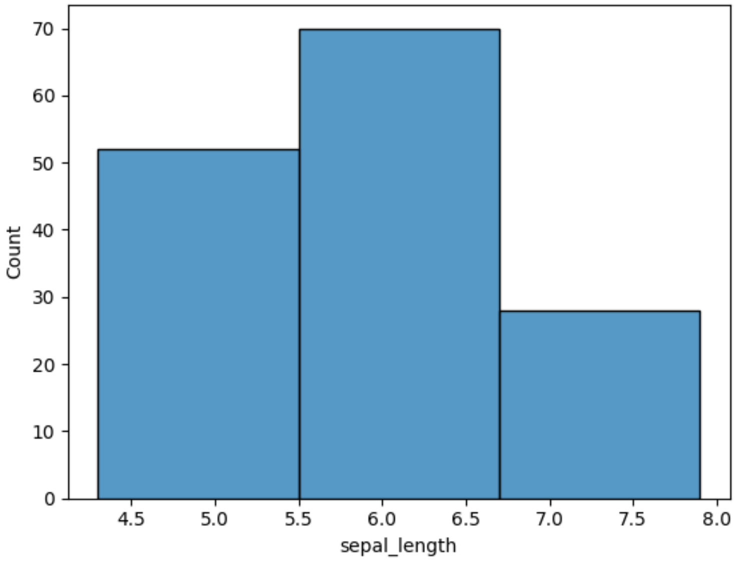
```
sns.scatterplot(data=a,x='sepal_length',y='petal_length',hue='species')
```

```
<Axes: xlabel='sepal_length', ylabel='petal_length'>
```



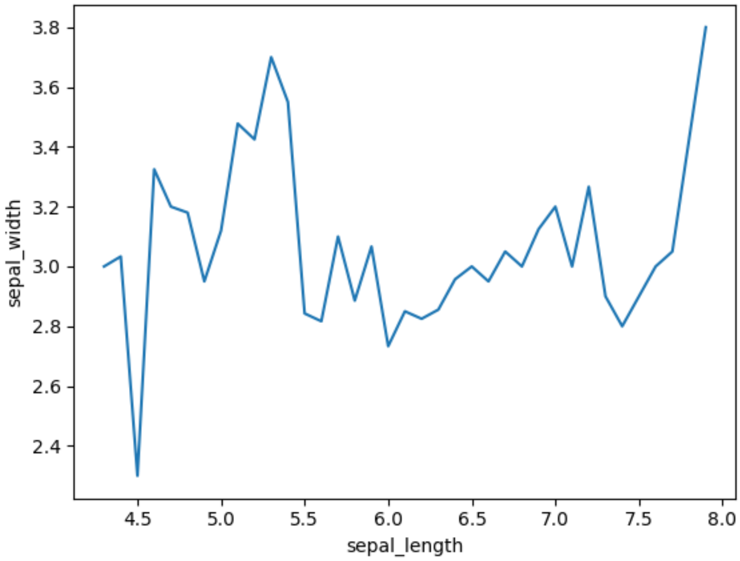
```
sns.histplot(data=a['sepal_length'],bins=3)
```

```
<Axes: xlabel='sepal_length', ylabel='Count'>
```



```
sns.lineplot(data=a,x='sepal_length',y='sepal_width',errorbar=None)
```

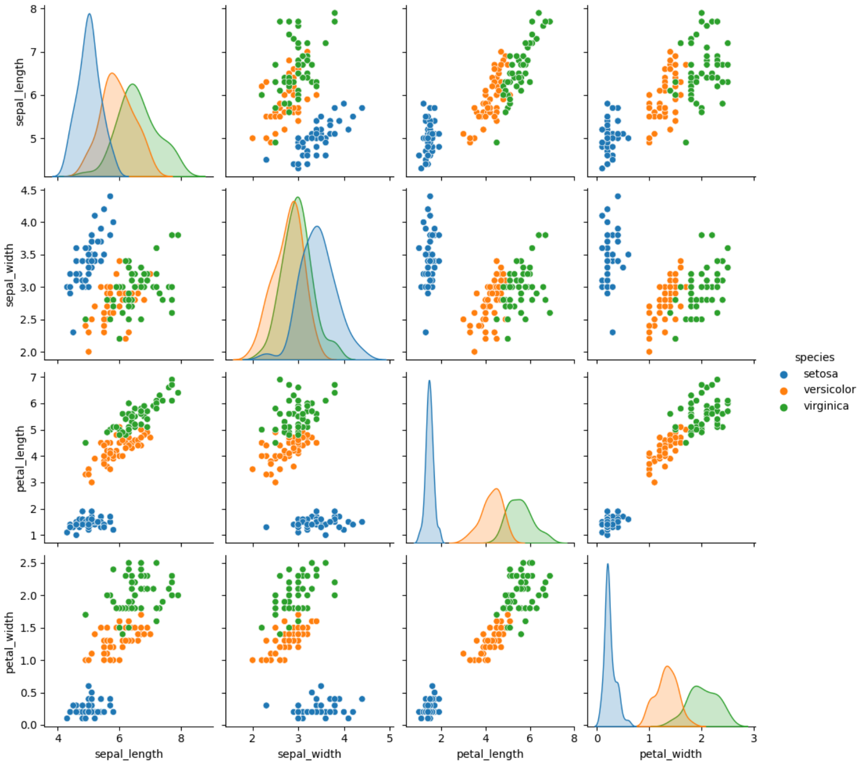
```
<Axes: xlabel='sepal_length', ylabel='sepal_width'>
```



Pairplot is created to view the relationship between each of the variable with others present in the data.

```
sns.pairplot(data=a,hue='species')
```

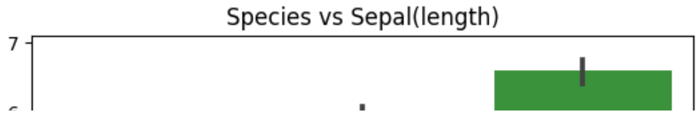
```
<seaborn.axisgrid.PairGrid at 0x7a7eecd3f4f0>
```



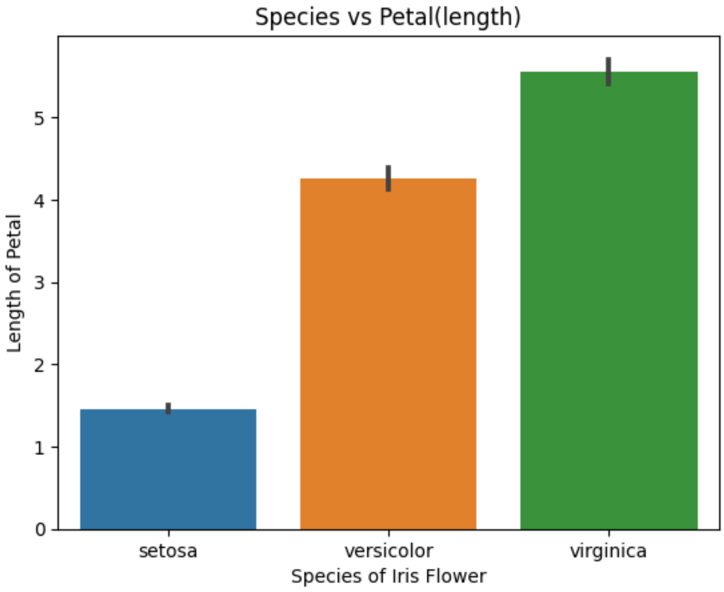
Data based on each species

```
import matplotlib.pyplot as plt
```

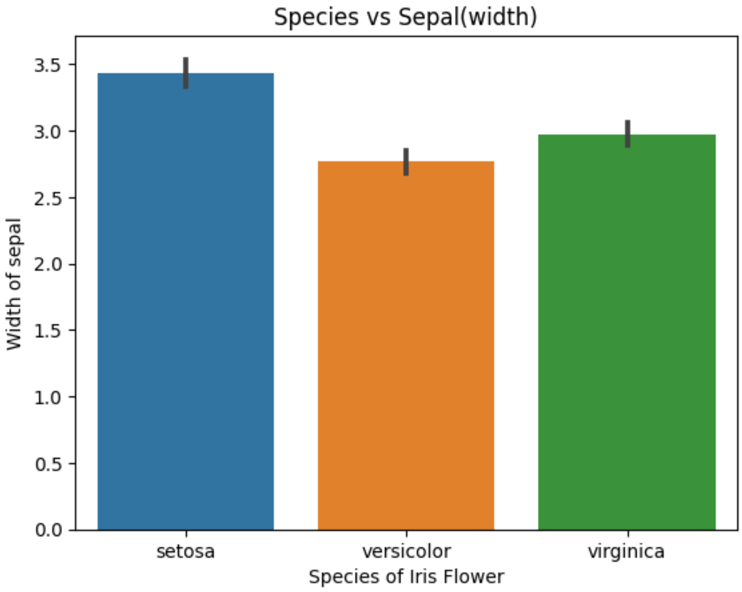
```
sns.barplot(data=a,x='species',y='sepal_length') # Virginica is the species with highest sepal length
plt.xlabel('Species of Iris Flower')
plt.ylabel('Length of sepal')
plt.title("Species vs Sepal(length)")
plt.show()
```



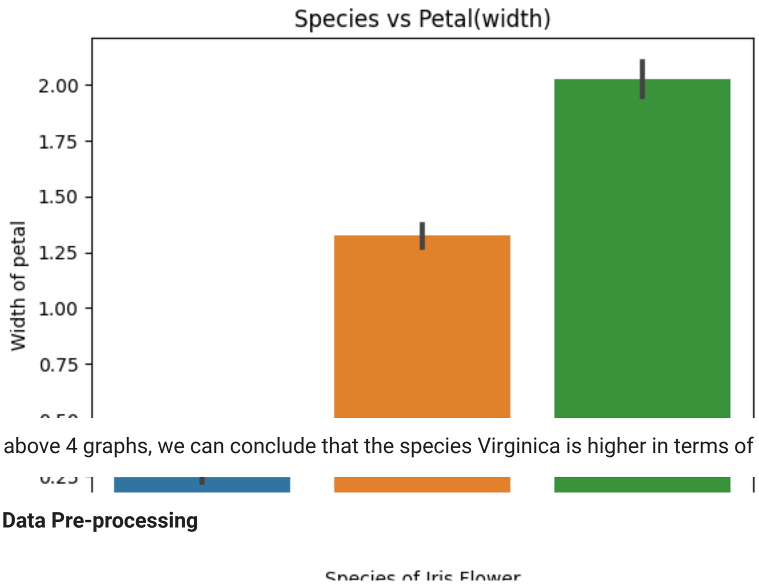
```
sns.barplot(data=a,x='species',y='petal_length')# Virginica is the species with highest petal length
plt.xlabel('Species of Iris Flower')
plt.ylabel('Length of Petal')
plt.title("Species vs Petal(length)")
plt.show()
```



```
sns.barplot(data=a,x='species',y='sepal_width') # Setosa is the species with highest sepal width
plt.xlabel('Species of Iris Flower')
plt.ylabel('Width of sepal')
plt.title("Species vs Sepal(width)")
plt.show()
```



```
sns.barplot(data=a,x='species',y='petal_width') # Virginica is the species with highest petal width
plt.xlabel('Species of Iris Flower')
plt.ylabel('Width of petal')
plt.title("Species vs Petal(width)")
plt.show()
```



From above 4 graphs, we can conclude that the species Virginica is higher in terms of size.

3. **Data Pre-processing**

Splitting the data into input and output

```
x=a.drop(columns='species') # x- input
y=a['species'] # y - output
```

Since, the data has to be classified use of categorical value is to be noted. So, to make easier, the 3 classes are converted into numerics.i.e, 'setosa'=1, 'versicolor'=2, 'virginica'=3.

```
y=y.replace({'setosa':1, 'versicolor':2, 'virginica':3})
```

Standardising the input data

```
from sklearn.preprocessing import StandardScaler

std=StandardScaler()

import pandas as pd

x=pd.DataFrame(data=std.fit_transform(x),columns=x.columns)
```

Data gets divided into training and testing data

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

4. **Model building**

As, the data is composed of multi class variable, 2 algorithms are used for model training - (1) K Nearest Classification; (2) Random Forests

(1) K Nearest Classification

```
from sklearn.neighbors import KNeighborsClassifier

kc=KNeighborsClassifier(n_neighbors=2) # defining a model

kc.fit(x_train,y_train) # training the model
```

```
▾      KNeighborsClassifier
KNeighborsClassifier(n_neighbors=2)
```

```
score_1=kc.score(x_train,y_train)
print('Accuracy Score for KNeighborsClassifier model with training data =',score_1)
score_2=kc.score(x_test,y_test)
print('Accuracy Score for KNeighborsClassifier model with test data =',score_2)
```

```
Accuracy Score for KNeighborsClassifier model with training data = 0.9916666666666667
Accuracy Score for KNeighborsClassifier model with test data = 0.8666666666666667
```

```
kc_a=KNeighborsClassifier(n_neighbors=5) # same model with increased number of neighbors
```

```
kc_a.fit(x_train,y_train)# training
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
score_1=kc_a.score(x_train,y_train)
print('Accuracy Score for KNeighborsClassifier model with training data =',score_1)
score_2=kc_a.score(x_test,y_test)
print('Accuracy Score for KNeighborsClassifier model with test data =',score_2)
```

```
Accuracy Score for KNeighborsClassifier model with training data = 0.9916666666666667
Accuracy Score for KNeighborsClassifier model with test data = 0.9
```

(2) Random Forests

```
from sklearn.ensemble import RandomForestClassifier
```

```
rfc=RandomForestClassifier(n_estimators=10) # building the model
```

```
rfc.fit(x_train,y_train)# training the model
```

```
▾        RandomForestClassifier
RandomForestClassifier(n_estimators=10)
```

```
rfc.fit(x_train,y_train)
scorea=rfc.score(x_train,y_train)
print('Accuracy Score for RandomForestClassifier model with training data =',scorea)
scoreb=rfc.score(x_test,y_test)
print('Accuracy Score for RandomForestClassifier model with test data =',scoreb)
```

```
Accuracy Score for RandomForestClassifier model with training data = 1.0
Accuracy Score for RandomForestClassifier model with test data = 0.9
```

```
rfc_a=RandomForestClassifier(n_estimators=50)
rfc_a.fit(x_train,y_train)
```

```
▾        RandomForestClassifier
RandomForestClassifier(n_estimators=50)
```

```
score_a=rfc_a.score(x_train,y_train)
print('Accuracy Score for RandomForestClassifier model with training data =',score_a)
score_b=rfc_a.score(x_test,y_test)
print('Accuracy Score for RandomForestClassifier model with test data =',score_b)
```

```
Accuracy Score for RandomForestClassifier model with training data = 1.0
Accuracy Score for RandomForestClassifier model with test data = 0.9
```

By comparing (1) and (2), when using random forest classifier,overfitting of model is observed. But the model "kc_a" built using KNeighborsClassifier algorithm with n_neighbors=5 is better, and hence selected for flower iris classification.

```
import pickle
```

```
pickle.dump(kc_a,open('/content/drive/MyDrive/ONE/iris_classification_model.pkl','wb'))
```

The model chosen for iris flower classification is saved using the module pickle in Google Drive which can be later used for the same.