OASIS INFOBYTE DATASCIENCE INTERNSHIP

TASK 1

JONNADA HANIRUDH REDDY

IRIS FLOWER CLASSIFICATION

In [20]:
```
#import main 2 libraries
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#bringing the csv file to a pandas dataframe
data=pd.read_csv('iris.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	5.0	3.6	1.4	0.2	setosa

In [4]:
```
data.shape
```

Out[4]:
```
(150, 6)
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0    150 non-null    int64
 1   Sepal.Length  150 non-null    float64
 2   Sepal.Width   150 non-null    float64
 3   Petal.Length  150 non-null    float64
 4   Petal.Width   150 non-null    float64
 5   Species       150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:
```
data['Species'].unique()
```

Out[6]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [7]:
```
#we don't need the 1st column so let's drop that
data=data.iloc[:,1:]
```

In [8]:
```
data.info()
```

```
<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
```

In [9]:
```
data.describe()
```

Out[9]:

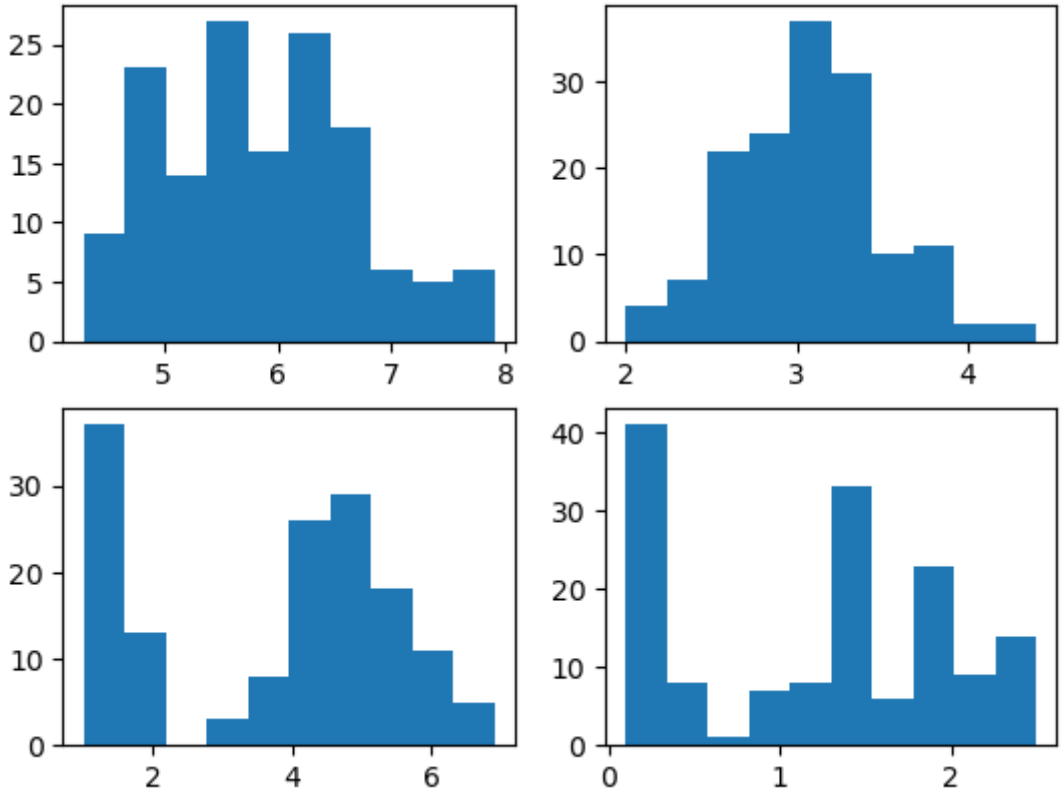
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [10]:
```
#check for null values
data.isnull().sum()
```

Out[10]:
```
Sepal.Length    0
Sepal.Width     0
Petal.Length    0
Petal.Width     0
Species         0
dtype: int64
```

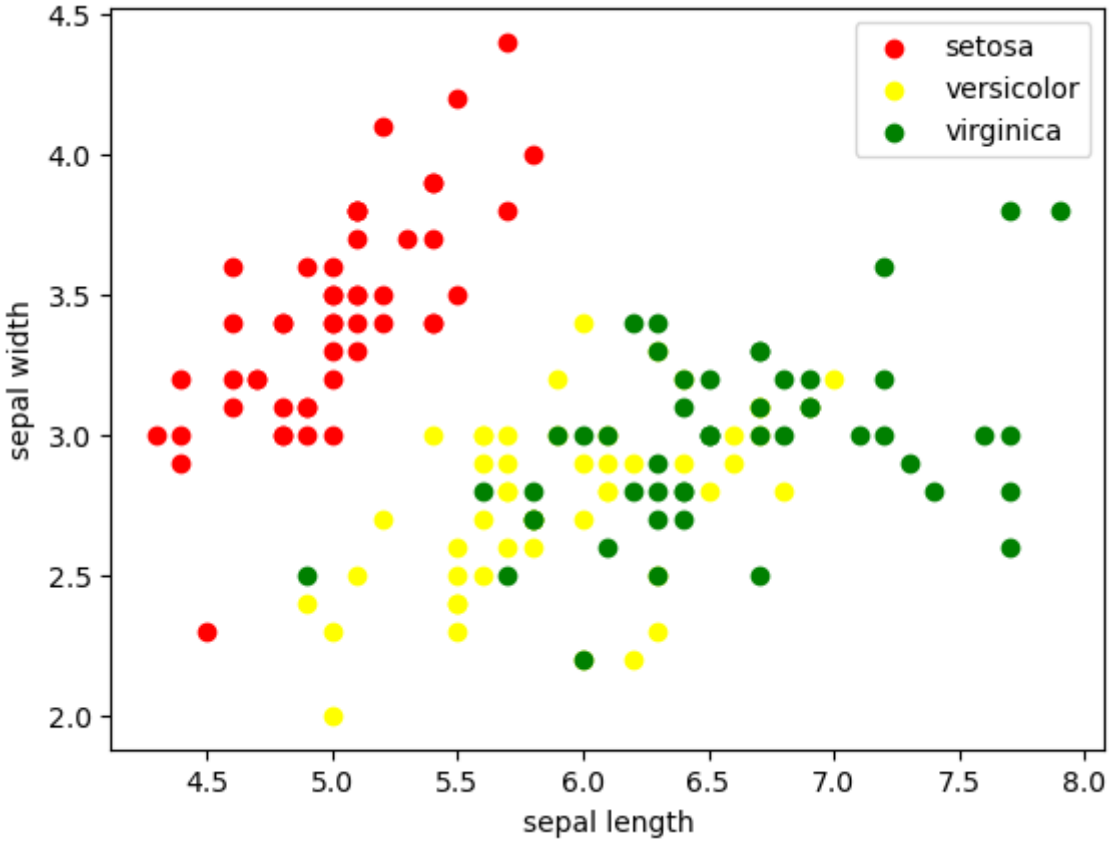
In [11]:
```
import matplotlib.pyplot as plt
```

In [13]:
```
#histograms
plt.subplot(2,2,1)
plt.hist(data['Sepal.Length'])
plt.subplot(2,2,2)
plt.hist(data['Sepal.Width'])
plt.subplot(2,2,3)
plt.hist(data['Petal.Length'])
plt.subplot(2,2,4)
plt.hist(data['Petal.Width'])
plt.show;
```

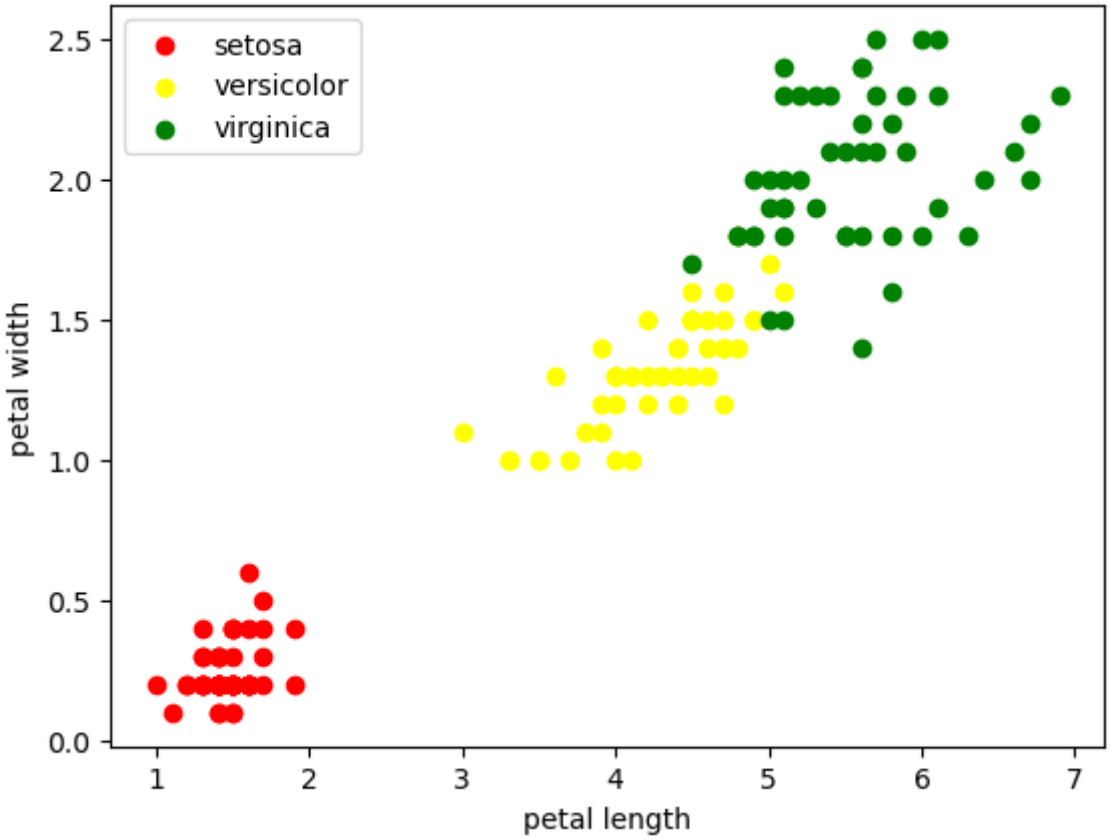


In [14]:
```
#scatter plots
colors=['red','yellow','green']
species=['setosa', 'versicolor', 'virginica']
```

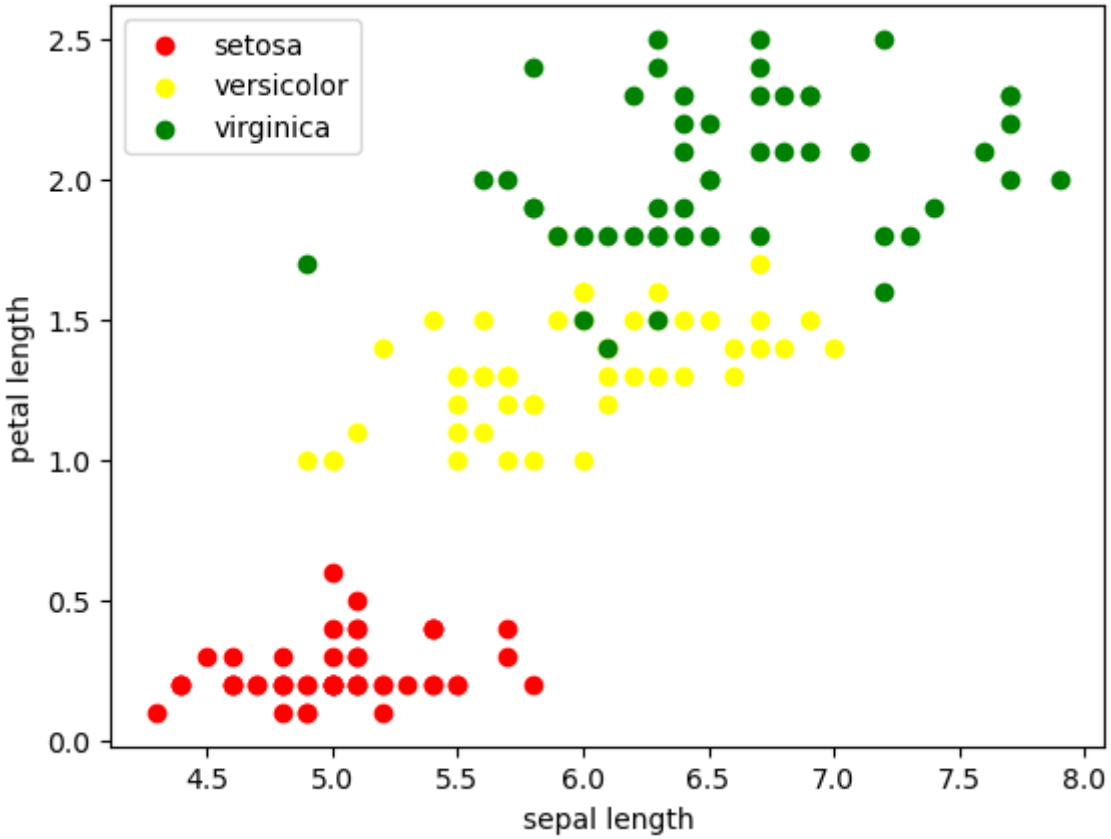
In [15]:
```
for i in range(3):
    x=data[data['Species']==species[i]]
    plt.scatter(x['Sepal.Length'],x['Sepal.Width'], c=colors[i], label=species[i])
plt.xlabel('sepal length')
plt.ylabel('sepal width')
plt.legend();
```



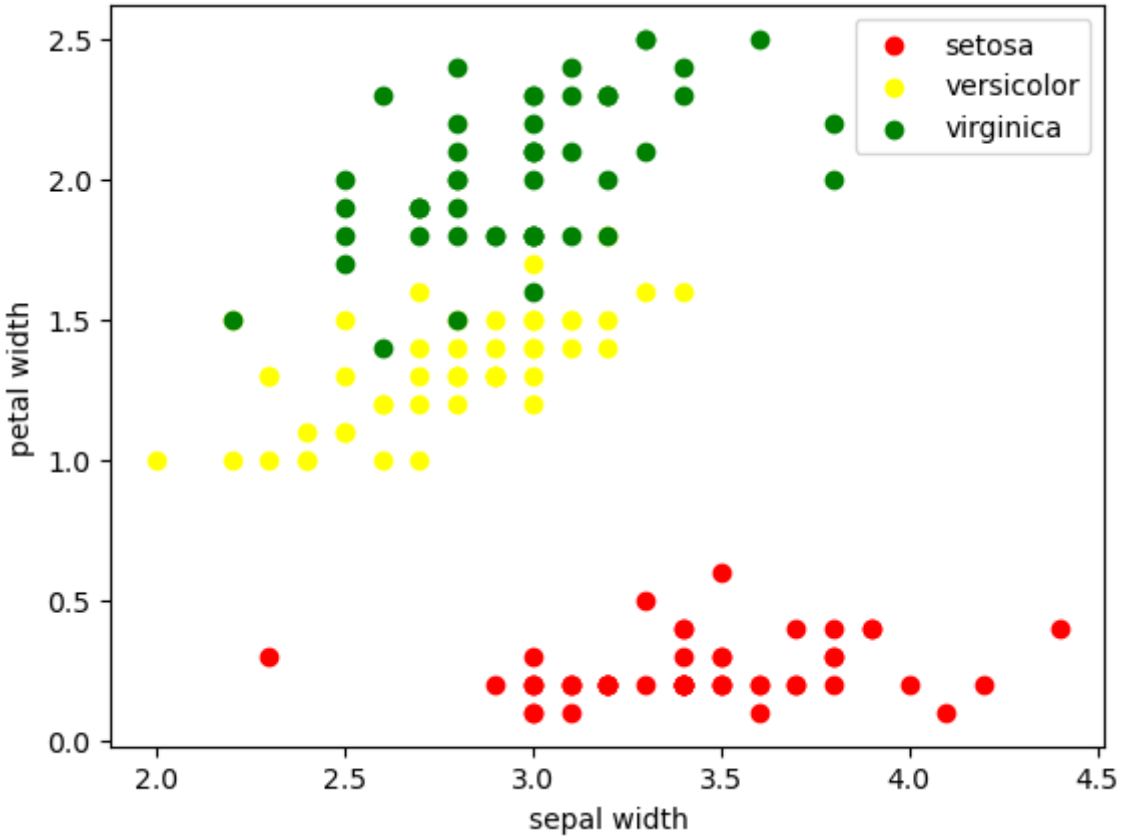
In [16]:
```
for i in range(3):
    x=data[data['Species']==species[i]]
    plt.scatter(x['Petal.Length'],x['Petal.Width'], c=colors[i], label=species[i])
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.legend();
```



In [17]:
```
for i in range(3):
    x=data[data['Species']==species[i]]
    plt.scatter(x['Sepal.Length'],x['Petal.Width'], c=colors[i], label=species[i])
plt.xlabel('sepal length')
plt.ylabel('petal length')
plt.legend();
```



In [18]:
```
for i in range(3):
    x=data[data['Species']==species[i]]
    plt.scatter(x['Sepal.Width'],x['Petal.Width'], c=colors[i], label=species[i])
plt.xlabel('sepal width')
plt.ylabel('petal width')
plt.legend();
```



In [21]:
```
#corealation matrix to check inter depandability of columns
data.corr()
```

Out[21]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.000000	-0.117570	0.871754	0.817941
Sepal.Width	-0.117570	1.000000	-0.428440	-0.366126
Petal.Length	0.871754	-0.428440	1.000000	0.962865
Petal.Width	0.817941	-0.366126	0.962865	1.000000

In [22]:
```
# as the output is a classification of strings we need to import label encoder to make that numeric
from sklearn.preprocessing import LabelEncoder
```

In [23]:
```
le= LabelEncoder()
```

In [24]:
```
data['Species']= le.fit_transform(data['Species'])
```

In [25]:
```
# 'setosa' == 0, 'versicolor' == 1, 'virginica' == 2
data['Species'].unique()
```

Out[25]:
```
array([0, 1, 2])
```

In [26]:
```
# separating inputs & outputs
x=data.drop(columns=['Species'])
y=data['Species']
```

In [27]:
```
x.head()
```

Out[27]:

	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

In [28]:
```
y.head()
```

Out[28]:
```
0    0
1    0
2    0
3    0
4    0
Name: Species, dtype: int32
```

In [29]:
```
# train test split  70% in train
from sklearn.model_selection import train_test_split
```

In [30]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.3)
```

In [31]:
```
x_train.head()
```

Out[31]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
9	4.9	3.1	1.5	0.1
44	5.1	3.8	1.9	0.4
137	6.4	3.1	5.5	1.8
101	5.8	2.7	5.1	1.9
58	6.6	2.9	4.6	1.3

In [32]:
```
y_test.head()
```

Out[32]:
```
51    1
50    1
7     0
43    0
94    1
Name: Species, dtype: int32
```

In [33]:
```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

In [34]:
```
column_trans=make_column_transformer((OneHotEncoder(sparse=False),[]),remainder='passthrough')
```

In [35]:
```
scaler=StandardScaler()
```

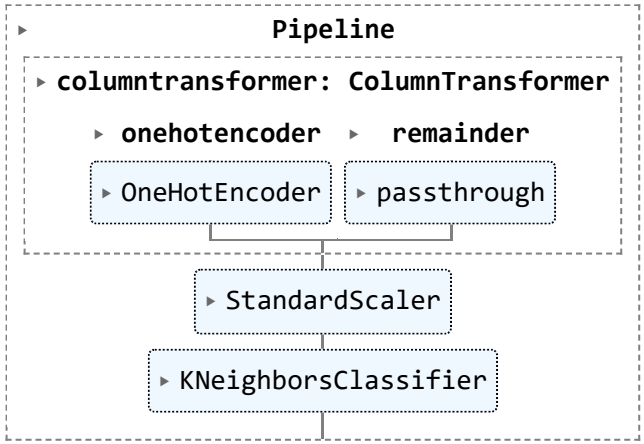
In [36]:
```
# KNN(k nearest neighbors) algorithm
from sklearn.neighbors import KNeighborsClassifier
```

In [37]:
```
knn= KNeighborsClassifier()
```

In [38]:
```
pipe = make_pipeline(column_trans,scaler,knn)
```

In [39]:
```
pipe.fit(x_train,y_train)
```

Out[39]:



In [40]:
```
y_pred_knn= pipe.predict(x_test)
```

In [41]:
```
r2_score(y_test,y_pred_knn)
```

Out[41]:
```
0.9075342465753424
```

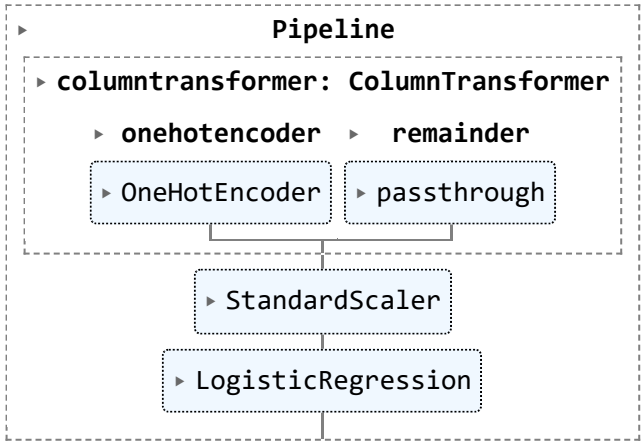
In [42]:
```
#logistic regression algorithm
from sklearn.linear_model import LogisticRegression
```

In [43]:
```
lr= LogisticRegression()
```

In [44]:
```
pipe=make_pipeline(column_trans,scaler,lr)
```

In [45]:
```
pipe.fit(x_train,y_train)
```

Out[45]:



In [46]:
```
y_pred_lr=pipe.predict(x_test)
```

In [47]:
```
r2_score(y_test,y_pred_lr)
```

Out[47]:
```
0.9691780821917808
```

In [48]:
```
#testing the pipe is predicting or not
pipe.predict([[1.2,1.5,1.6,1.2]])
```

Out[48]:
```
array([0])
```

In [49]:
```
# saving the model
import pickle
```

In [50]:
```
pickle.dump(pipe,open('iris_flower.pkl','wb'))
```

In [ ]: