In [86]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [87]:

```
df=pd.read_csv("C:/Users/91740/OneDrive/Desktop/INTERNSHIP+PROJECT/archive (1)/Iris.csv")
df.head()
```

Out[87]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [88]:

```
#Delete the id column
df=df.drop(columns=['Id'])
df.head()
```

Out[88]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [89]:

```
#To display ststsistics about data
df.describe()
```

Out[89]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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 [90]:

```
#To display basic info about data type
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [91]:

```
#To disply no.of samples on each class
df['Species'].value_counts()
```

Out[91]:

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

```
In [92]: #PREPROCESSING THE DATASET
```

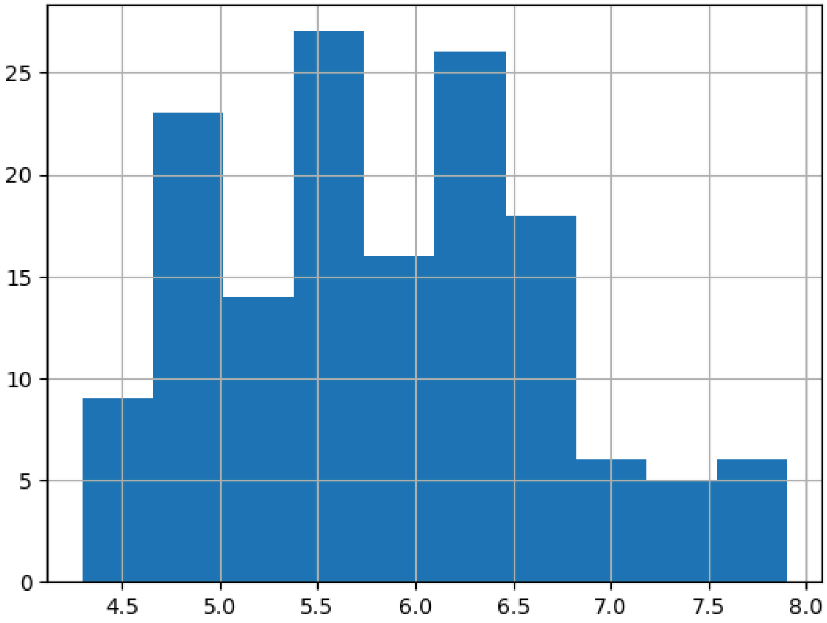
```
In [93]: #check the null values
         df.isnull().sum()
```

```
Out[93]: SepalLengthCm    0
         SepalWidthCm     0
         PetalLengthCm    0
         PetalWidthCm     0
         Species          0
         dtype: int64
```

```
In [94]: #EXPLORATORY DATA ANALYSIS
```

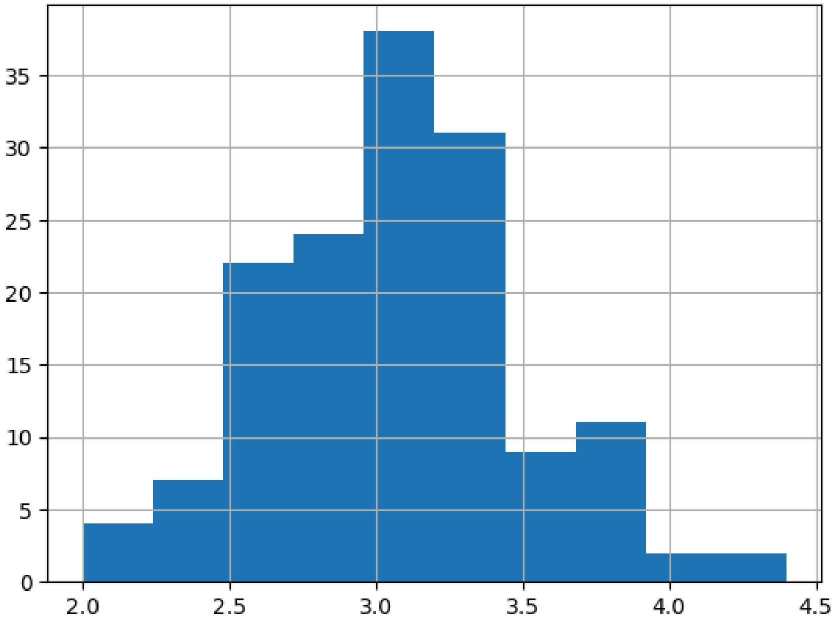
```
In [95]: #histogram for all the classes
         df['SepalLengthCm'].hist()
```

```
Out[95]: <Axes: >
```



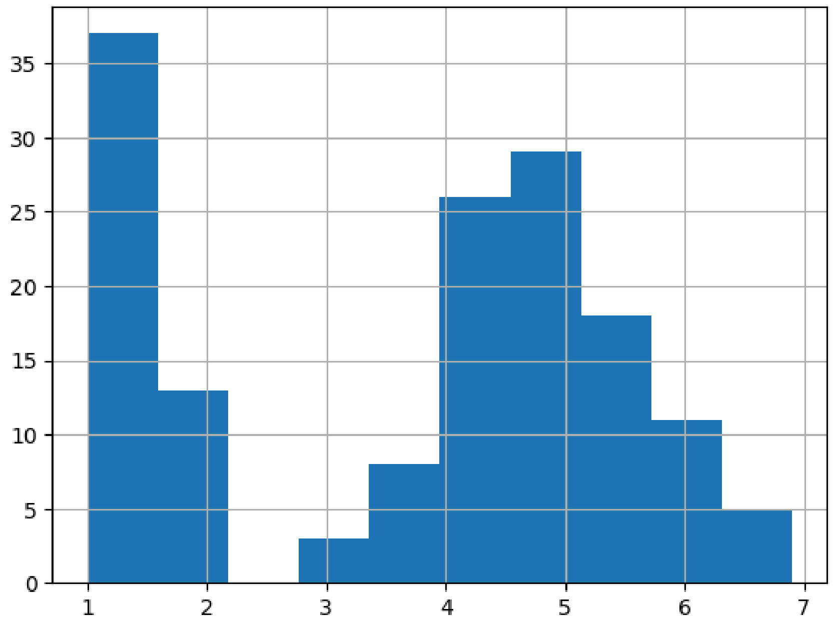
```
In [96]: df['SepalWidthCm'].hist()
```

```
Out[96]: <Axes: >
```



```
In [97]: df['PetalLengthCm'].hist()
```

```
Out[97]: <Axes: >
```

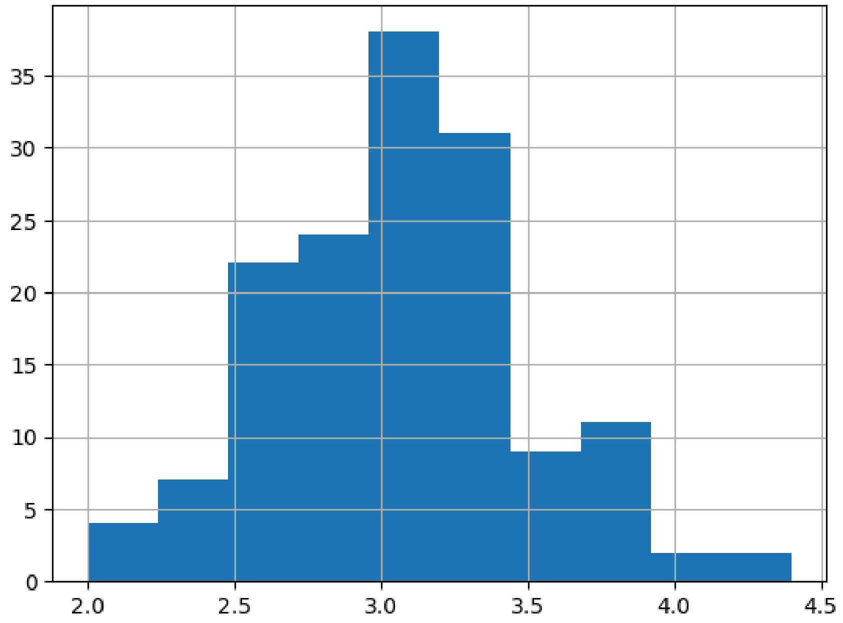


In [98]:

```
df['SepalWidthCm'].hist()
```

Out[98]:

```
<Axes: >
```



In [99]:

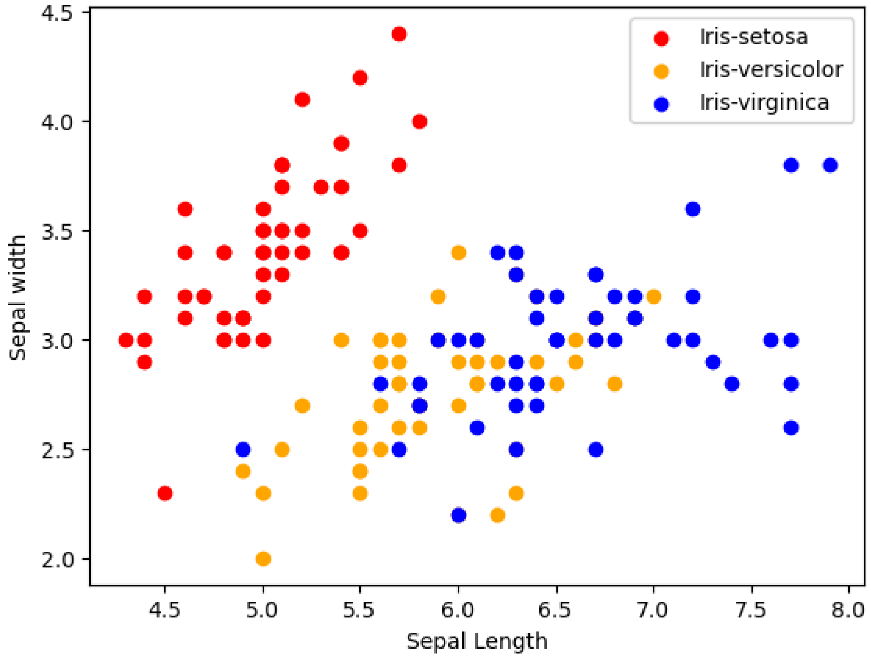
```
#scatterplot
colors=['red','Orange','blue']
species=['Iris-setosa','Iris-versicolor','Iris-virginica']
```

In [100…

```
#Sepal Length vs Sepal Width
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'],x['SepalWidthCm'],c=colors[i],label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal width")
plt.legend()
```

Out[100]:

```
<matplotlib.legend.Legend at 0x23700d472d0>
```

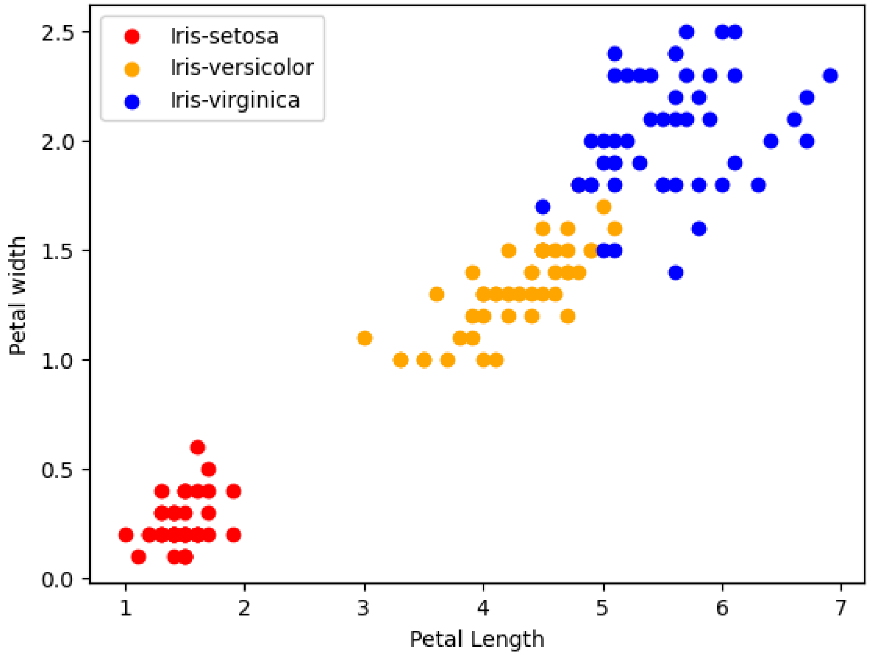


In [101…

```
#Petal Length vs Petal Width
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['PetalLengthCm'],x['PetalWidthCm'],c=colors[i],label=species[i])
plt.xlabel("Petal Length")
plt.ylabel("Petal width")
plt.legend()
```

Out[101]:

```
<matplotlib.legend.Legend at 0x237018bd610>
```

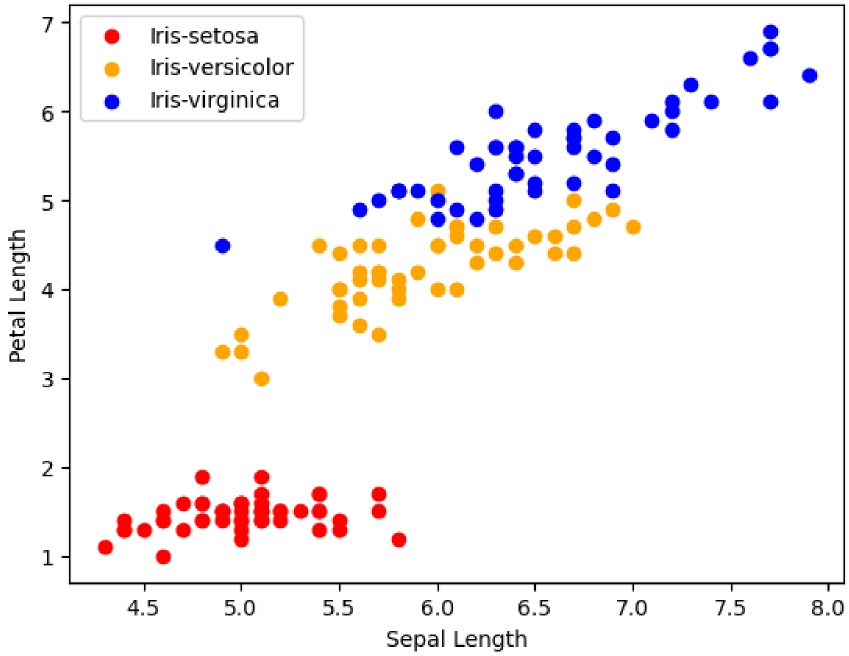


In [102…

```
#Sepal Length vs Petal Length
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'],x['PetalLengthCm'],c=colors[i],label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.legend()
```

Out[102]:

```
<matplotlib.legend.Legend at 0x237021fee10>
```

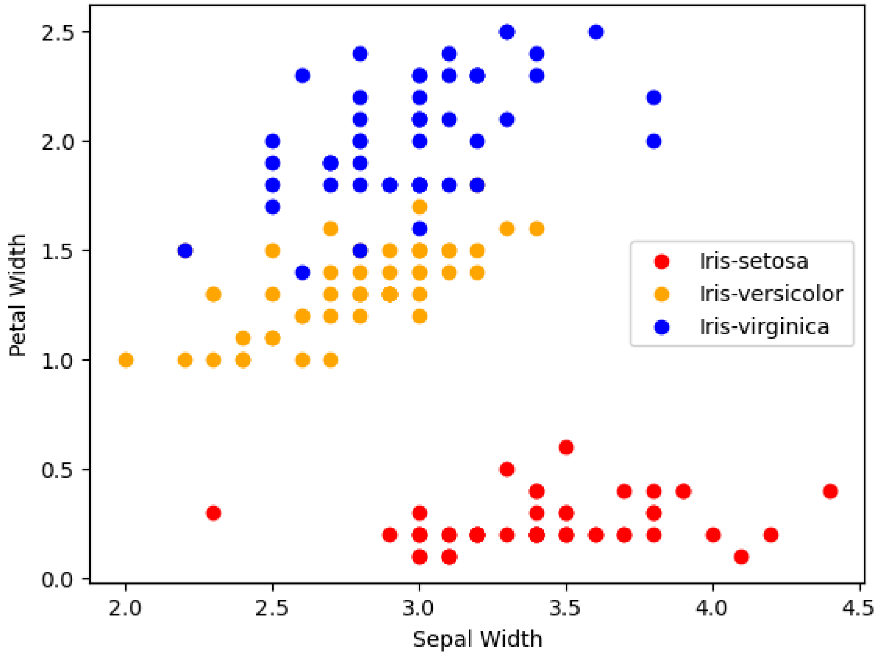


In [103…

```
#Petal Width vs Sepal Width
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalWidthCm'],x['PetalWidthCm'],c=colors[i],label=species[i])
plt.xlabel("Sepal Width")
plt.ylabel("Petal Width")
plt.legend()
```

Out[103]:

```
<matplotlib.legend.Legend at 0x237026a0d90>
```



In [104…

```
df1=df.drop(columns=['Species'])
df1.head()
```

Out[104]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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 [105…

```
#COORELATION MATRIX
df1.corr()
```

Out[105]:

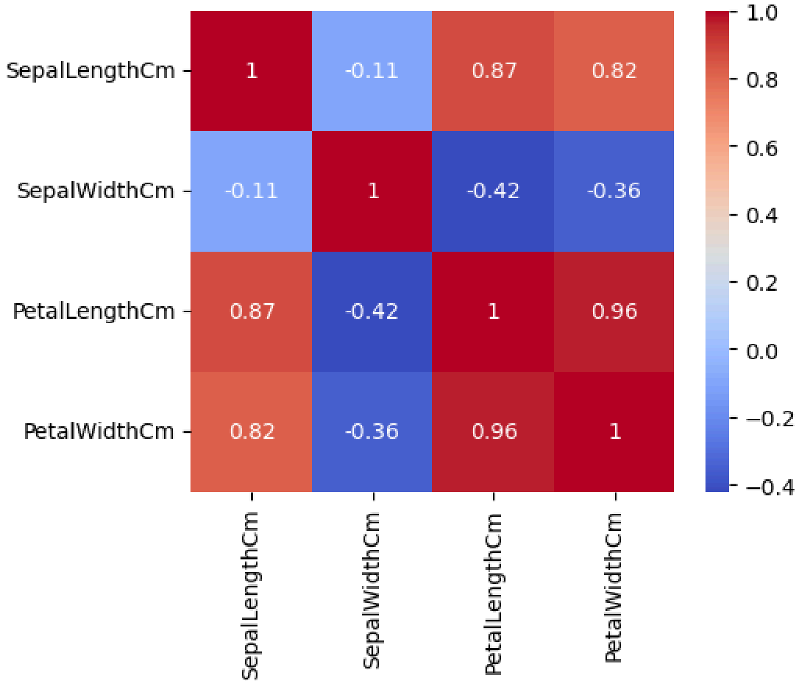
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [106…

```
corr=df1.corr()
fig,ax=plt.subplots(figsize=(5,4))
sns.heatmap(corr,annot=True,ax=ax,cmap='coolwarm')
```

Out[106]:

```
<Axes: >
```



In [123…

```
#LABEL ENCODER
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [124…

```
df['Species']=le.fit_transform(df['Species'])
```

In [125…

```
df.head()
```

Out[125]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [126…

```
#MODEL TRAINING #train-70%,test-30%
from sklearn.model_selection import train_test_split
x=df.drop(columns=['Species'])
y=df['Species']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=30)
```

In [127…

```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [128…

```
#model trainig
model.fit(x_train,y_train)
```

Out[128]:

```
▾ LogisticRegression
LogisticRegression()
```

In [129…

```
#print metric to get performance
print("Accuracy:",model.score(x_test,y_test)*100) #In percentage format
```

```
Accuracy: 96.66666666666667
```

In [130…

```
#K-NN NEIGHBOUR
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
```

In [131…

```
model.fit(x_train,y_train)
```

Out[131]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [132…

```
#print metric to get performance
print("Accuracy:",model.score(x_test,y_test)*100) #In percentage format
```

```
Accuracy: 93.33333333333333
```

In [133…

```
#DECISSION TREE CLASSIFIER
from sklearn.tree import DecisionTreeRegressor
model=DecisionTreeRegressor()
```

In [134…

```
model.fit(x_train,y_train)
```

Out[134]:

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [135…

```
#print metric to get performance
print("Accuracy:",model.score(x_test,y_test)*100) #In percentage format
```

```
Accuracy: 93.75
```

In [136…

```
#Save the model
import pickle
filename='saved_model.sav'
pickle.dump(model,open(filename,'wb'))
```

In [137…

```
load_model=pickle.load(open(filename,'rb'))
```

In [138…

```
load_model.predict([[6.0,2.2,4.0,1.0]]) # (0= Iris-Setosa, 1= Iris-versicolor, 2= Iris-virginica)
```

```
C:\Users\91740\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature name
s, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
```

Out[138]:

```
array([1.])
```

In [ ]:

In [ ]: