In [47]:
```
import pandas as pd
import numpy as np
import pickle
import warnings
warnings.filterwarnings('ignore')
```

In [48]:
```
data=pd.read_csv("C:/Users/gonab/OneDrive/Desktop/flower data.csv")
data
```

Out[48]:

	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
...	...	...	...	...	...
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

150 rows × 5 columns

In [49]:
```
data.describe()
```

Out[49]:

	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 [50]:
```
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 [51]:
```
data.head()
```

Out[51]:

	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 [52]:
```
data.tail()
```

Out[52]:

	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 [53]:
```
data.shape
```

Out[53]:
```
(150, 5)
```

In [54]:
```
data['species'].value_counts()
```

Out[54]:
```
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

In [55]:
```
dict_species={'setosa':0,'versicolor':1,'virginica':2,}
data.replace({'species':dict_species},inplace=True)
data
```

Out[55]:

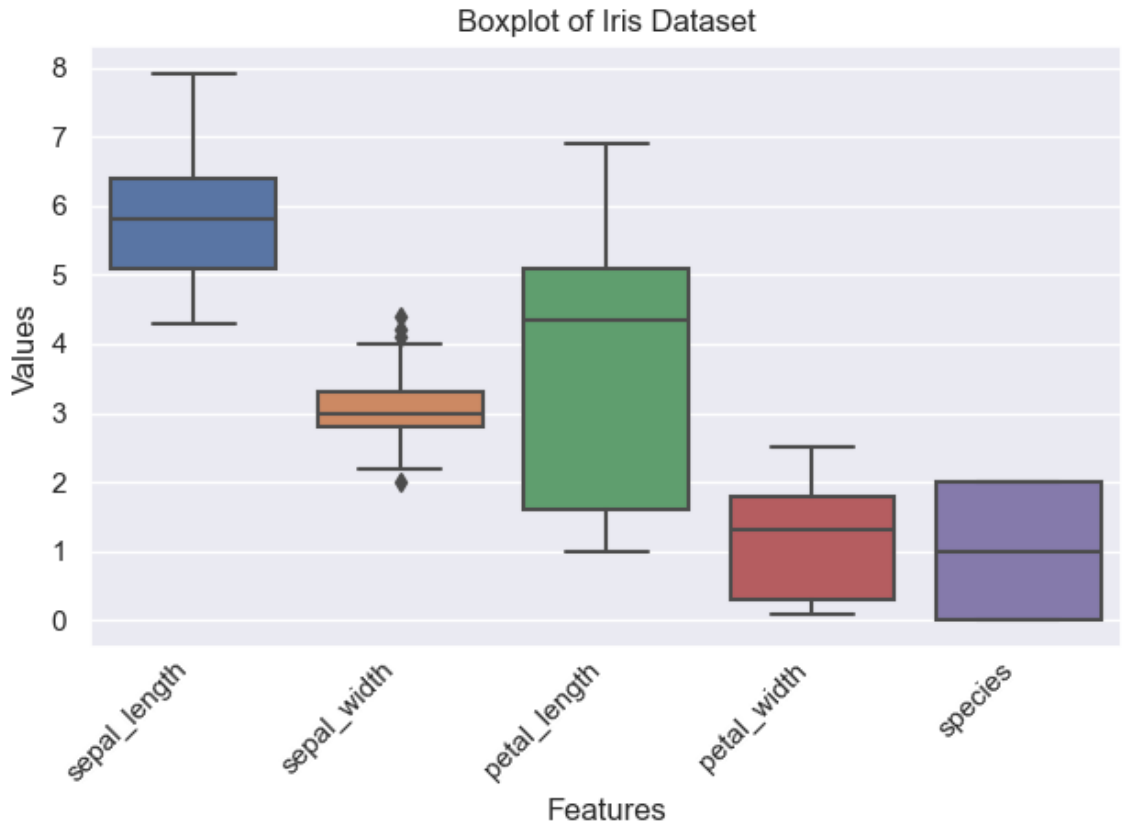
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

Data Visualization

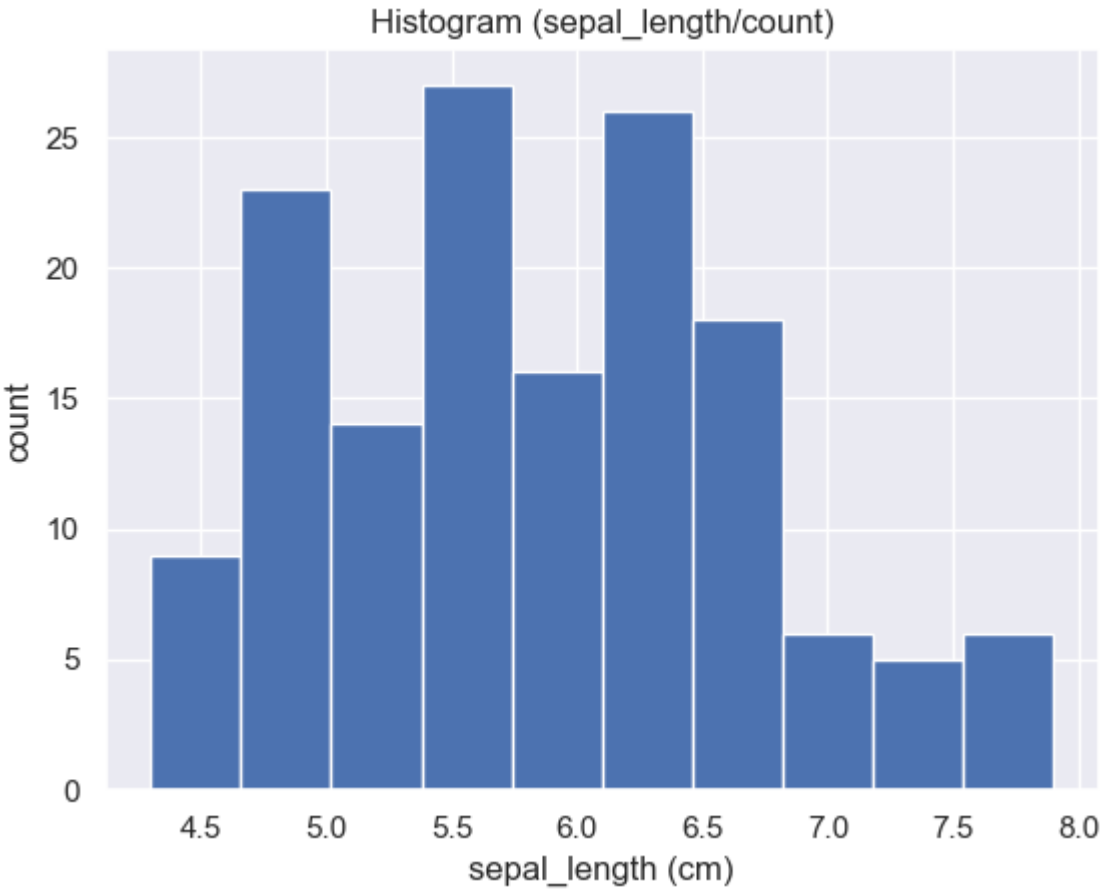
In [56]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [57]:
```
sns.boxplot(data=data)

plt.tick_params(axis='x', which='both', length=0)
plt.xticks(rotation=45, ha='right')

# Add labels and title
plt.xlabel('Features')
plt.ylabel('Values')
plt.title('Boxplot of Iris Dataset')

# Display the plot
plt.tight_layout()  # Ensures labels are not cut off
plt.show()
```



In [58]:

```
sns.set()
plt.hist(data['sepal_length'])
plt.xlabel('sepal_length (cm)')
plt.ylabel('count')
plt.title('Histogram (sepal_length/count)')
plt.show()
```



In [59]:
```
x=data.drop(['species'],axis=1)
x
```

Out[59]:

	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 × 4 columns

In [60]:
```
y=data['species']
y
```

Out[60]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: species, Length: 150, dtype: int64
```

Data Splitting into Training data and Testing data

In [61]:
```
from sklearn.model_selection import train_test_split
x_test,x_train,y_test,y_train=train_test_split(x,y,test_size=0.3,random_sta
x_test.head(10)
```

Out[61]:

	sepal_length	sepal_width	petal_length	petal_width
81	5.5	2.4	3.7	1.0
133	6.3	2.8	5.1	1.5
137	6.4	3.1	5.5	1.8
75	6.6	3.0	4.4	1.4
109	7.2	3.6	6.1	2.5
96	5.7	2.9	4.2	1.3
105	7.6	3.0	6.6	2.1
66	5.6	3.0	4.5	1.5
0	5.1	3.5	1.4	0.2
122	7.7	2.8	6.7	2.0

In [62]:
```
x_train.head()
```

Out[62]:

	sepal_length	sepal_width	petal_length	petal_width
73	6.1	2.8	4.7	1.2
18	5.7	3.8	1.7	0.3
118	7.7	2.6	6.9	2.3
78	6.0	2.9	4.5	1.5
76	6.8	2.8	4.8	1.4

In [63]:
```
y_test.head()
```

Out[63]:
```
81     1
133    2
137    2
75     1
109    2
Name: species, dtype: int64
```

In [64]:
```
y_train.head()
```

Out[64]:
```
73     1
18     0
118    2
78     1
76     1
Name: species, dtype: int64
```

In [65]:
```
x_train.shape
```

Out[65]:
```
(45, 4)
```

In [66]:
```
x_test.shape
```

Out[66]:
```
(105, 4)
```

In [67]:
```
y_train.shape
```

Out[67]:
```
(45,)
```

In [68]:
```
y_test.shape
```

Out[68]:
```
(105,)
```

Random Forest

In [69]:
```
#from sklearn.model_selection import GridSearchCV #GridSearchCV is for para
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor(random_state=0)
reg.fit(x_train,y_train)
```

Out[69]:
```
RandomForestRegressor(random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [70]:
```
y_pred=reg.predict(x_test)
y_pred
```

Out[70]:
```
array([0.97, 1.51, 1.99, 1.  , 2.  , 0.98, 2.  , 0.98, 0.  , 2.  , 1.  ,
       0.  , 0.  , 0.  , 0.97, 1.84, 0.  , 0.  , 0.  , 0.98, 0.  , 0.98,
       2.  , 0.  , 1.  , 1.95, 0.  , 1.58, 1.99, 1.  , 0.97, 2.  , 0.97,
       0.  , 0.97, 1.99, 0.  , 0.  , 0.97, 1.95, 0.  , 2.  , 0.  , 0.  ,
       1.58, 0.96, 2.  , 1.51, 1.51, 2.  , 0.97, 0.  , 0.  , 1.5 , 1.99,
       0.  , 0.  , 0.  , 1.36, 2.  , 0.  , 2.  , 2.  , 0.  , 1.  , 1.  ,
       1.58, 1.  , 2.  , 0.  , 2.  , 1.  , 2.  , 0.98, 0.97, 1.  , 0.  ,
       1.  , 0.97, 0.  , 0.98, 1.99, 2.  , 0.  , 0.97, 2.  , 1.51, 0.  ,
       2.  , 0.  , 1.36, 1.99, 1.99, 0.98, 1.99, 1.  , 1.  , 1.81, 1.99,
       0.  , 1.  , 1.41, 0.03, 1.  , 2.  ])
```

Accuracy for Random Forest

In [71]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[71]:
```
0.9481385135135135
```

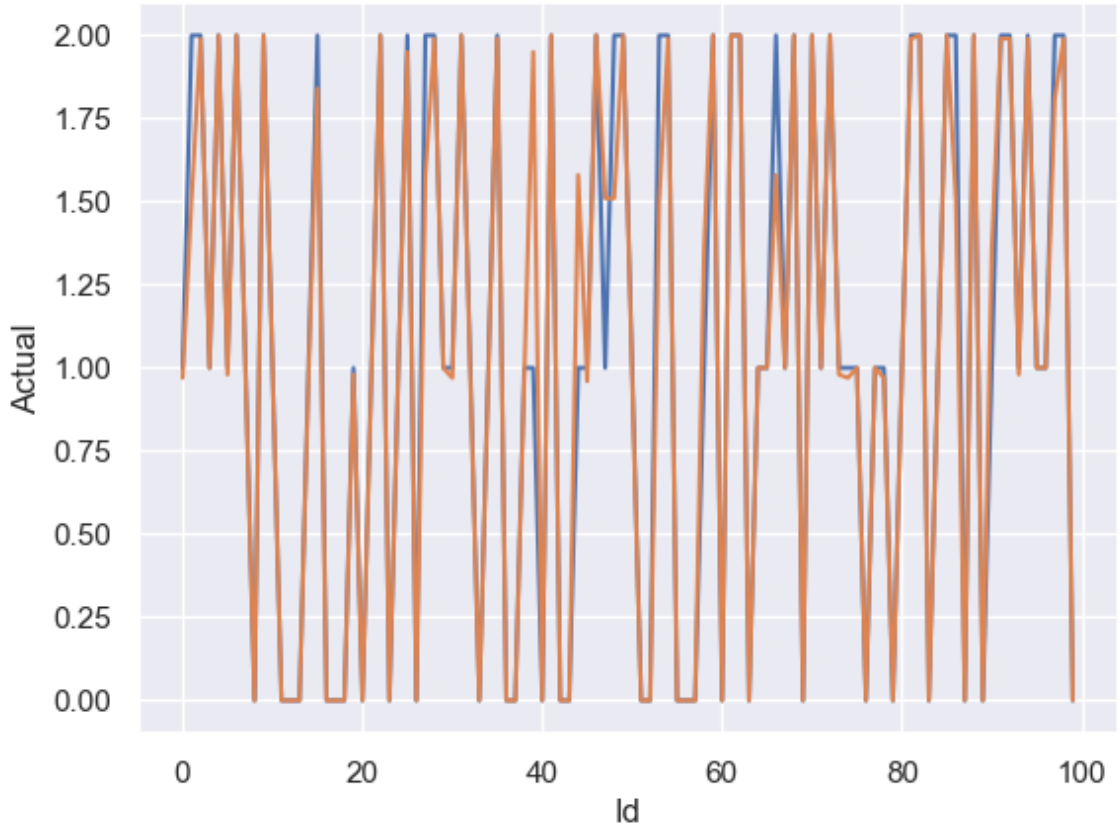
In [72]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[72]:

	index	Actual	Predicted	Id
0	81	1	0.97	0
1	133	2	1.51	1
2	137	2	1.99	2
3	75	1	1.00	3
4	109	2	2.00	4
5	96	1	0.98	5
6	105	2	2.00	6
7	66	1	0.98	7
8	0	0	0.00	8
9	122	2	2.00	9

In [73]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='Actual',data=Results.head(100))
sns.lineplot(x='Id',y='Predicted',data=Results.head(100))
plt.plot()
```

Out[73]: []



In [ ]:

In [ ]:

In [ ]: