OASIS INFOBYTE DATA SCIENCE INTERN

Ranjeeta Kumari

TASK - 1 IRIS FLOWER CLASSIFICATION

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
```

In [4]:

```
df = pd.read_csv(r'C:\Users\ajeet singh\OneDrive\Desktop\Iris.csv')
print(df)
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
df.tail()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [7]:

```
df.shape
```

Out[7]:

```
(150, 6)
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [14]:

```
df.dtypes
```

Out[14]:

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [16]:

```
df.describe()
```

Out[16]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [17]:

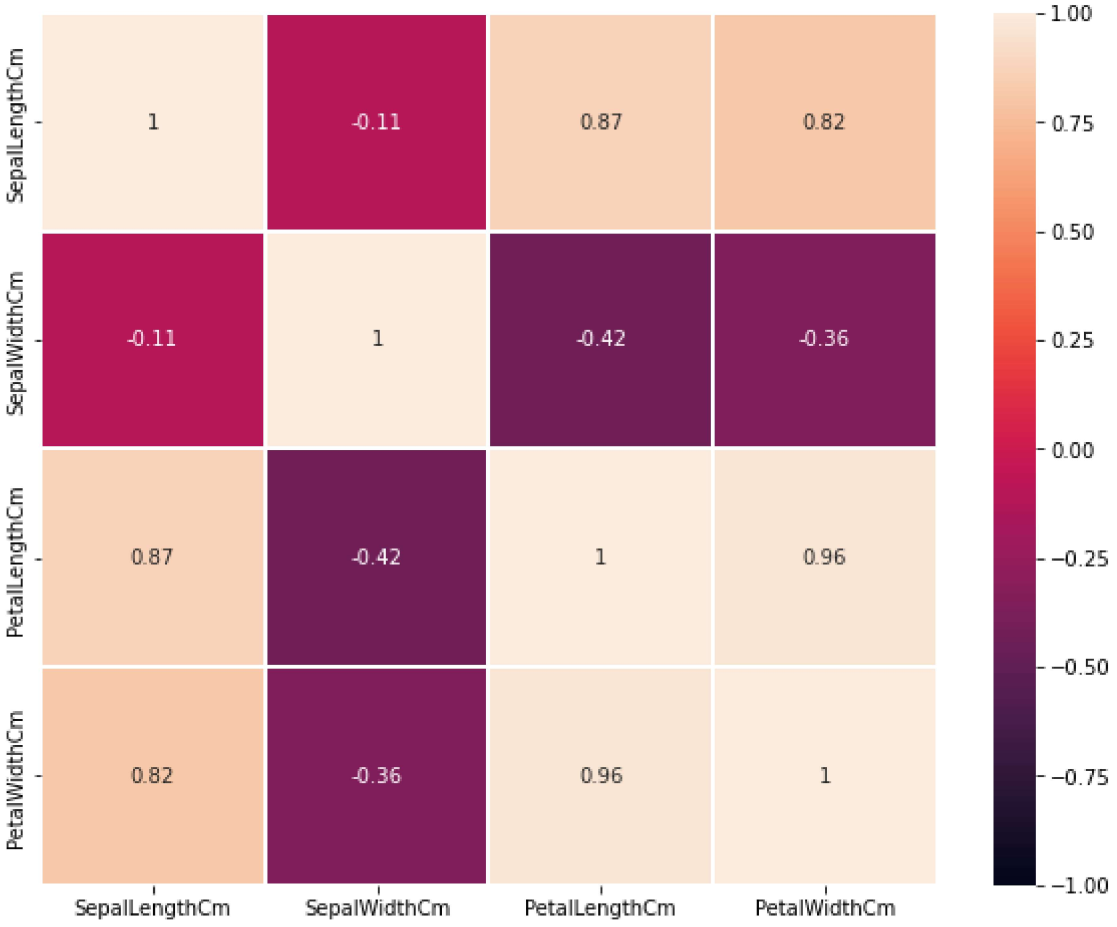
```
df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']].corr()
```

Out[17]:

	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 [13]:

```
plt.figure(figsize=(10,8))
sns.heatmap(df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
plt.show()
```



In [18]:

```
df.groupby('Species').describe()
```

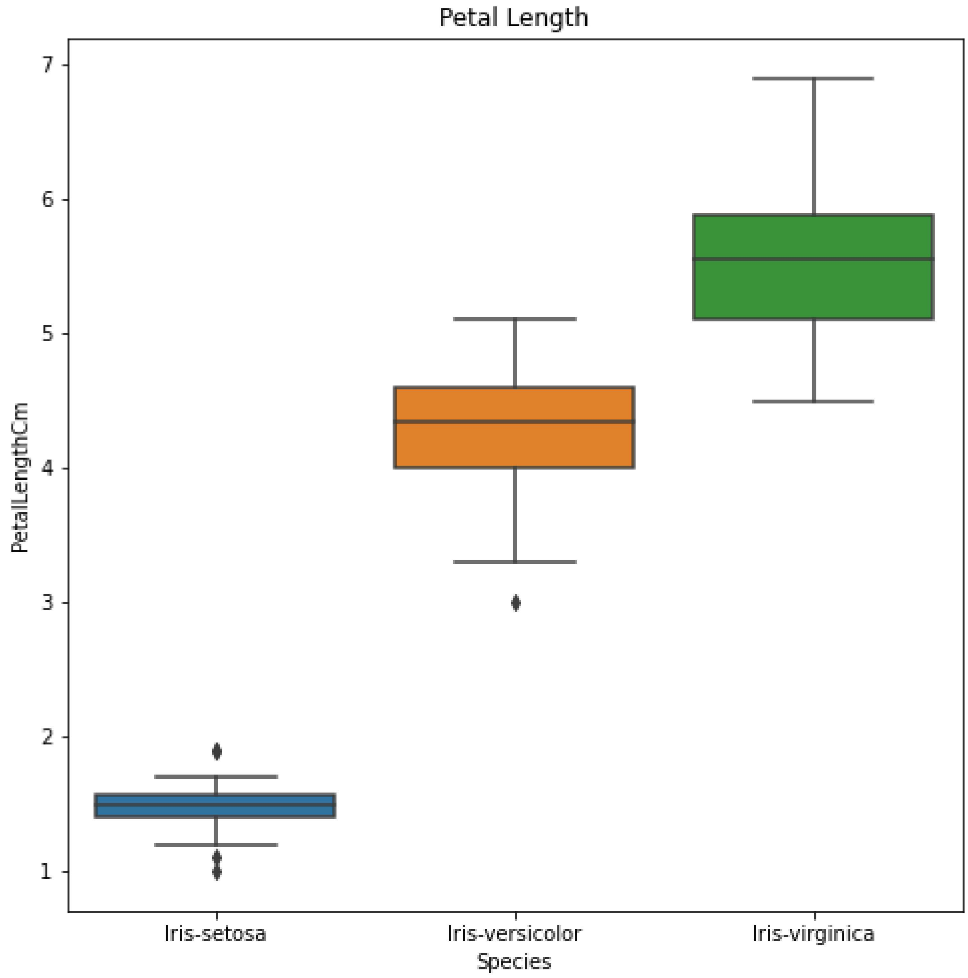
Out[18]:

	Id								SepalLengthCm		...	PetalLe
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%
Species												
Iris-setosa	50.0	25.5	14.57738	1.0	13.25	25.5	37.75	50.0	50.0	5.006	...	1.575
Iris-versicolor	50.0	75.5	14.57738	51.0	63.25	75.5	87.75	100.0	50.0	5.936	...	4.600
Iris-virginica	50.0	125.5	14.57738	101.0	113.25	125.5	137.75	150.0	50.0	6.588	...	5.875

3 rows × 40 columns

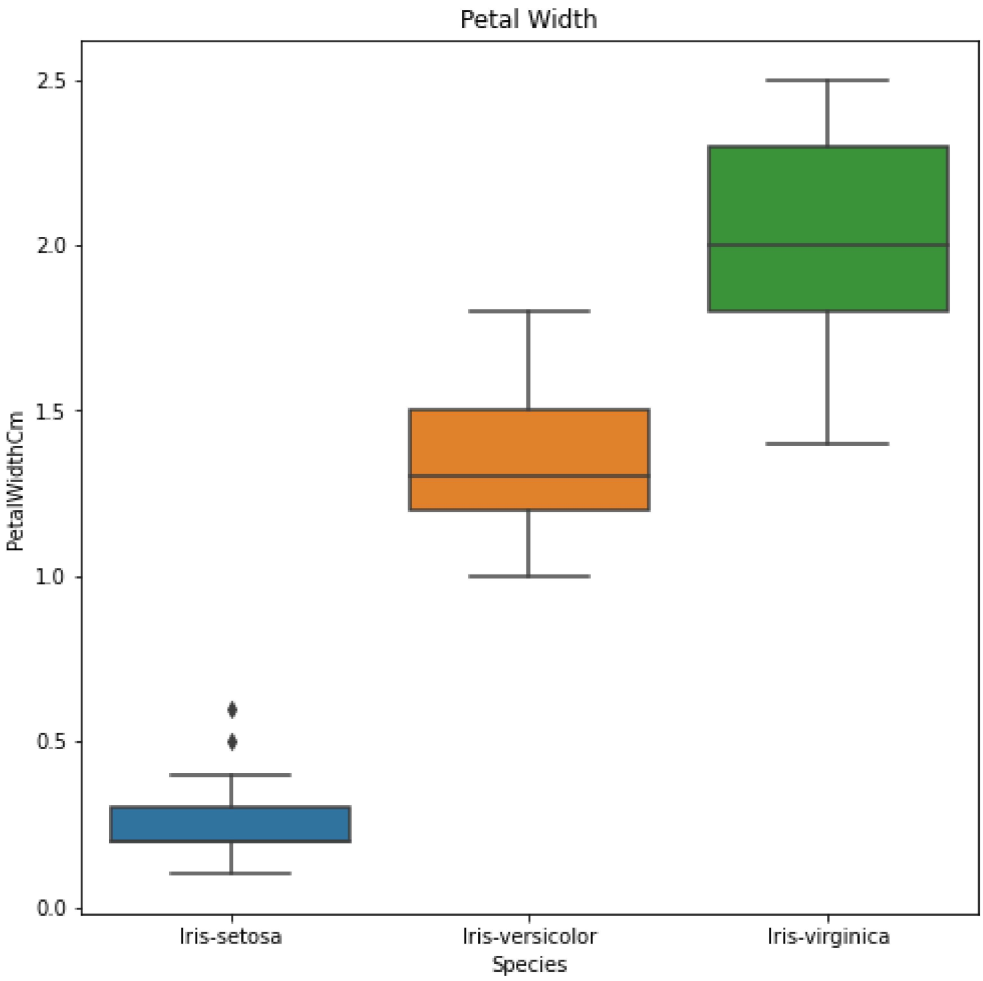
In [19]:

```
plt.figure(figsize=(8,8))
ax = sns.boxplot(x="Species", y="PetalLengthCm", data=df).set_title('Petal Length'
plt.show()
```



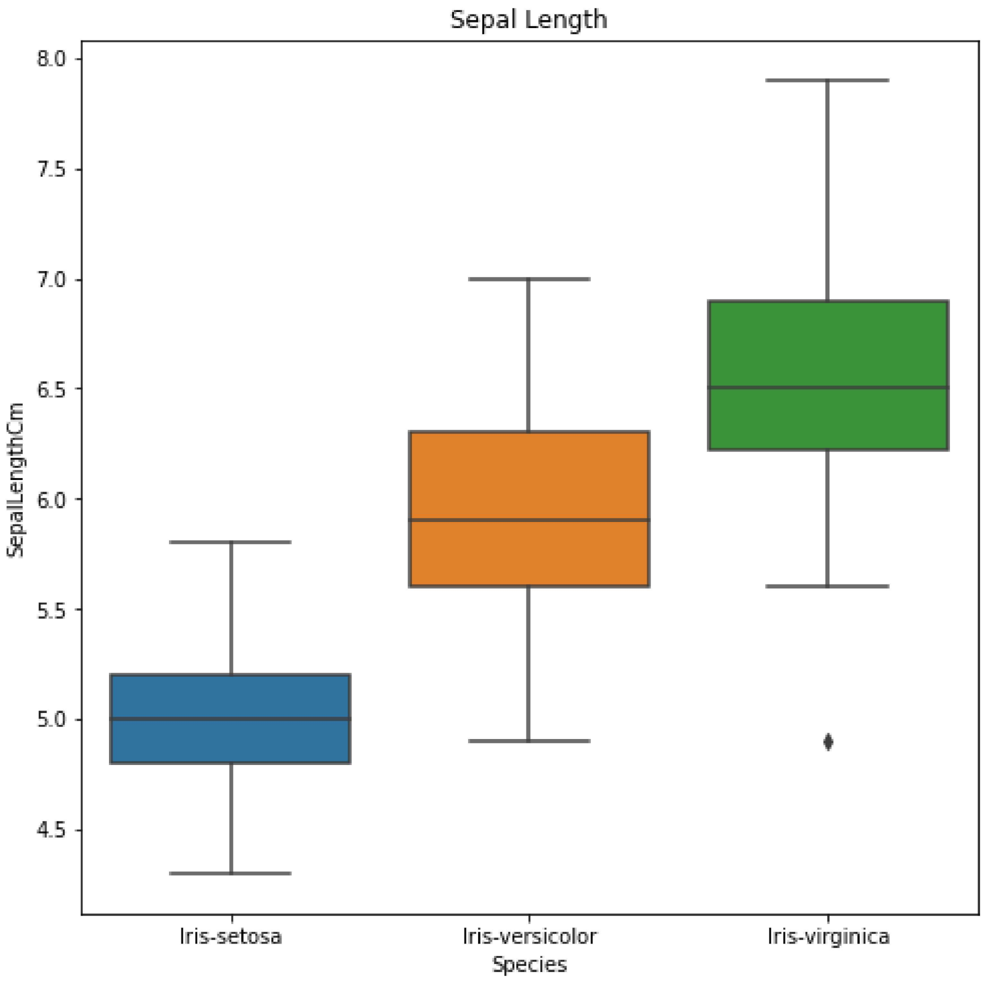
In [21]:

```
plt.figure(figsize=(8,8))
ax = sns.boxplot(x="Species", y="PetalWidthCm", data=df).set_title('Petal Width')
plt.show()
```



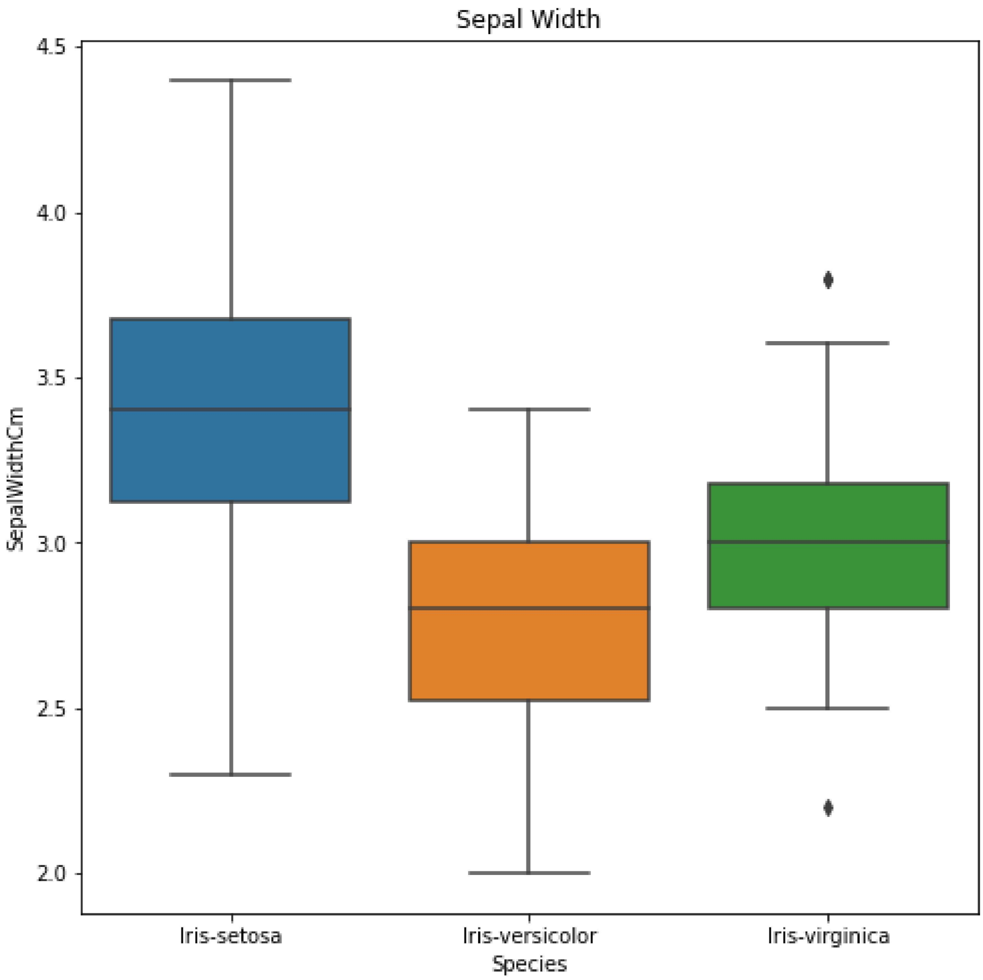
In [22]:

```
plt.figure(figsize=(8,8))
ax = sns.boxplot(x="Species", y="SepalLengthCm", data=df).set_title('Sepal Length'
plt.show()
```



In [23]:

```
plt.figure(figsize=(8,8))
ax = sns.boxplot(x="Species", y="SepalWidthCm", data=df).set_title('Sepal Width')
plt.show()
```



In [24]:

```
# We take 80% of data into training, and 20% into test
# For each set, a third belonds to each type of Iris
df.drop(['Id'], axis=1, inplace=True)
training = pd.concat([df[:40], df[50:90], df[100:140]])
test = pd.concat([df[40:50], df[90:100], df[140:]])
training_X = training[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWid
training_y = training['Species']
test_X  = test[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
test_y = test['Species']
```

In [25]:

```
print('Training set:', training_X.shape)
print('Test set:', test_X.shape)
```

```
Training set: (120, 4)
Test set: (30, 4)
```

In [26]:

```
from sklearn.linear_model import LogisticRegression
LR_classifier = LogisticRegression(solver='lbfgs', multi_class='multinomial', max_
print('Training accuracy:', LR_classifier.score(training_X, training_y))
print('Test accuracy:', LR_classifier.score(test_X, test_y))
```

```
Training accuracy: 0.975
Test accuracy: 1.0
```

In [27]:

```
from sklearn.neighbors import KNeighborsClassifier
KNN_classifier = KNeighborsClassifier().fit(training_X, training_y)
print('Training accuracy:', KNN_classifier.score(training_X, training_y))
print('Test accuracy:', KNN_classifier.score(test_X, test_y))
```

```
Training accuracy: 0.9666666666666667
Test accuracy: 1.0
```

In [28]:

```
from sklearn.svm import LinearSVC
SVC_classifier = LinearSVC(multi_class='crammer_singer', max_iter=3000).fit(traini
SVC_classifier.score(training_X, training_y)
print('Training accuracy:', SVC_classifier.score(training_X, training_y))
print('Test accuracy:', SVC_classifier.score(test_X, test_y))
```

```
Training accuracy: 0.975
Test accuracy: 1.0
```

```
D:\anacondaa1\lib\site-packages\sklearn\svm\_base.py:1206: ConvergenceWarning: Lib
linear failed to converge, increase the number of iterations.
  warnings.warn(
```

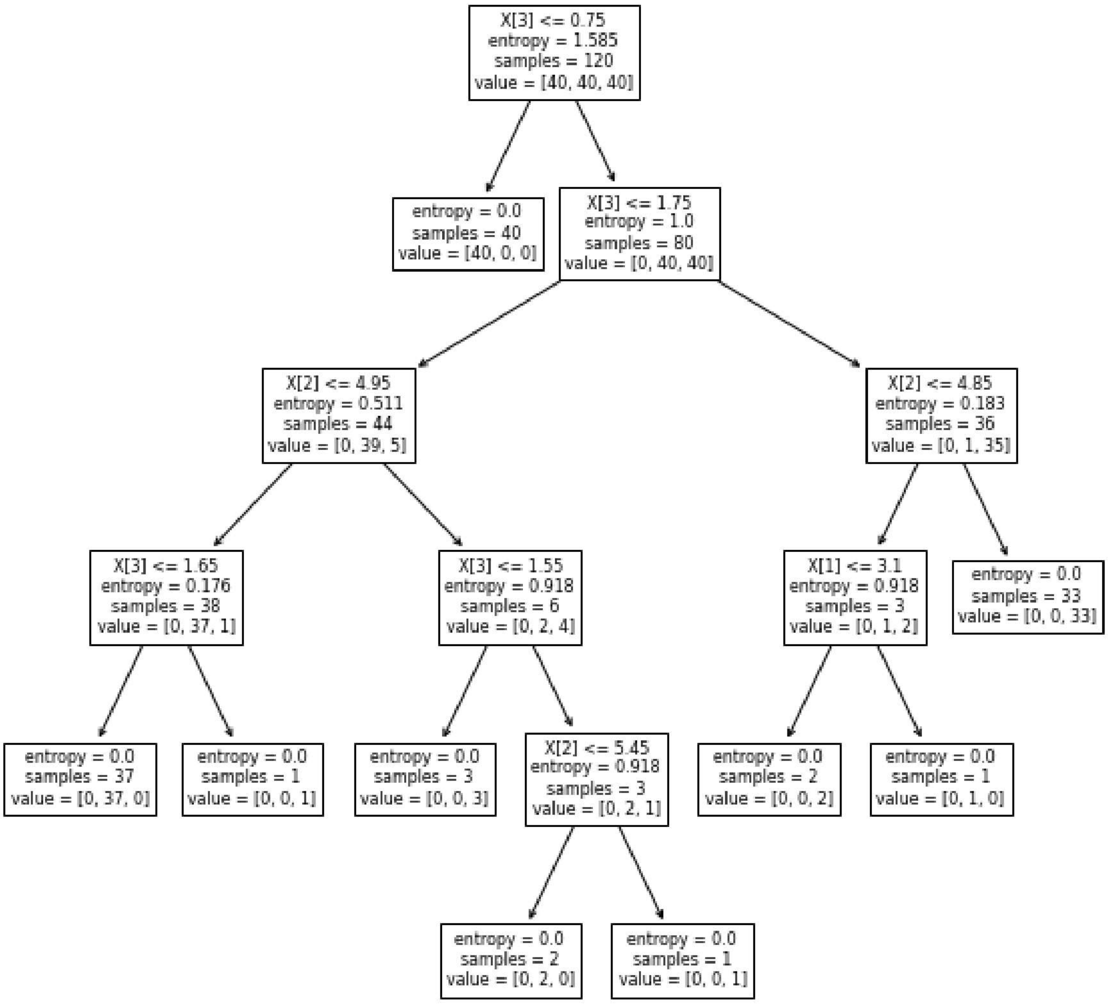
In [29]:

```
from sklearn.tree import DecisionTreeClassifier
dTree_classifier = DecisionTreeClassifier(criterion="entropy").fit(training_X, tra
print('Training accuracy:', dTree_classifier.score(training_X, training_y))
print('Test accuracy:', dTree_classifier.score(test_X, test_y))
```

```
Training accuracy: 1.0
Test accuracy: 1.0
```

In [30]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(10,10))
plot_tree(dTree_classifier)
plt.show()
```



In [ ]: