6) Aim: Write a Python program to implement Logistic Regression and plot the graphs.

In [113…
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
```

With Iris Dataset

In [3]:
```
df = pd.read_csv("Iris.csv")
```

In [4]:
```
df
```

Out[4]:

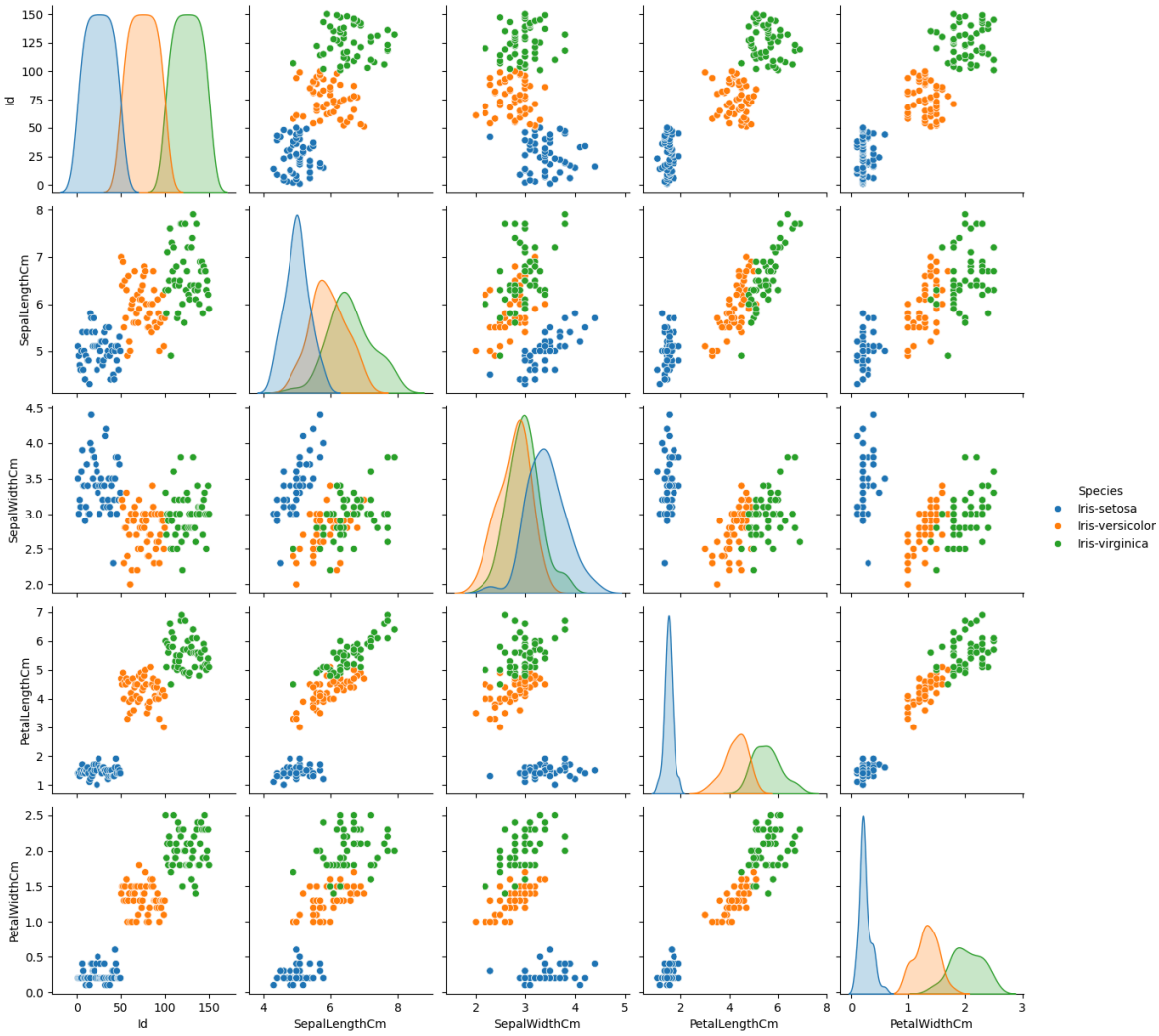
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	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

150 rows × 6 columns

In [5]:
```
import seaborn as sns
```

In [6]:
```
sns.pairplot(df,hue='Species')
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x228b754a610>
```



In [7]:
```
X = df.iloc[:,1:5]
Y = df.iloc[:,-1]
```

In [8]:
```
X
```

Out[8]:

	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
...	...	...	...	...
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 [9]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random
```

In [10]:
```
from sklearn.linear_model import LogisticRegression
dt=LogisticRegression()
```

In [11]:
```
dt.fit(X_train,Y_train)
```

Out[11]:
```
LogisticRegression()
```

In [12]:
```
Y_pred = dt.predict(X_test)
Y_pred
```

Out[12]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa'], dtype=object)
```

In [13]:
```
dt.score(X_test,Y_test)
```

Out[13]: 1.0

In [14]:
```
import numpy as np
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
```

In [15]:
```
print('Accuracy: %.3f' % accuracy_score(Y_test, Y_pred))
print('f1 score: %.3f' % f1_score(Y_test, Y_pred,average='micro'))
print('recall: %.3f' % recall_score(Y_test, Y_pred, average='macro'))
print('Precision: %.3f' % precision_score(Y_test, Y_pred,average='macro'))
```

```
Accuracy: 1.000
f1 score: 1.000
recall: 1.000
Precision: 1.000
```

In [16]:
```
import numpy as np

def callfuct():
    user_input = []

    SepalLengthCm = float(input("Enter value between 4.300000 to 7.900000 for Se
    SepalWidthCm = float(input("Enter value between 2.000000 to 4.400000  for Se
    PetalLengthCm = float(input("Enter value between 1.000000 to 6.900000  for P
    PetalWidthCm = float(input("Enter value between 0.100000 to 2.500000  for Pe
    user_input.append([SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm]

    user_input = np.array(user_input)

    predicted_classes = dt.predict(user_input)

    print(predicted_classes)
```

In [17]:
```
callfuct()
```

```
['Iris-versicolor']
C:\Users\nayan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\base.py:493: UserWarning: X does not have valid feature names, but LogisticRegre
ssion was fitted with feature names
  warnings.warn(
```

With Mushrooms Dataset

In [18]:
```
df1 = pd.read_csv("mushrooms.csv")
```

In [19]:
```
df1
```

Out[19]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...
3	p	x	y	w	t	p	f	c	n	n	...
4	e	x	s	g	f	n	f	w	b	k	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	e	k	s	n	f	n	a	c	b	y	...
8120	e	x	s	n	f	n	a	c	b	y	...
8121	e	f	s	n	f	n	a	c	b	n	...
8122	p	k	y	n	f	y	f	c	n	b	...
8123	e	x	s	n	f	n	a	c	b	y	...

8124 rows × 23 columns

In [20]:
```
df1.isnull().sum()
```

Out[20]:
```
class                     0
cap-shape                 0
cap-surface               0
cap-color                 0
bruises                   0
odor                      0
gill-attachment           0
gill-spacing              0
gill-size                 0
gill-color                0
stalk-shape               0
stalk-root                0
stalk-surface-above-ring  0
stalk-surface-below-ring  0
stalk-color-above-ring    0
stalk-color-below-ring    0
veil-type                 0
veil-color                0
ring-number               0
ring-type                 0
spore-print-color         0
population                0
habitat                   0
dtype: int64
```

In [28]:
```
missing_count = df1.isin(['?']).sum()
missing_count
```

Out[28]:
```
class                          0
cap-shape                      0
cap-surface                    0
cap-color                      0
bruises                        0
odor                           0
gill-attachment                0
gill-spacing                   0
gill-size                      0
gill-color                     0
stalk-shape                    0
stalk-root                  2480
stalk-surface-above-ring       0
stalk-surface-below-ring       0
stalk-color-above-ring         0
stalk-color-below-ring         0
veil-type                      0
veil-color                     0
ring-number                    0
ring-type                      0
spore-print-color              0
population                     0
habitat                        0
dtype: int64
```

In [30]:
```
# Replace '?' with NaN
data = df1.replace('?', pd.NA)

# Impute missing values with mode
data = df1.apply(lambda col: col.fillna(col.mode()[0]))
data
```

Out[30]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	p	x	s	n	t	p	f	c	n	k	...
1	e	x	s	y	t	a	f	c	b	k	...
2	e	b	s	w	t	l	f	c	b	n	...
3	p	x	y	w	t	p	f	c	n	n	...
4	e	x	s	g	f	n	f	w	b	k	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	e	k	s	n	f	n	a	c	b	y	...
8120	e	x	s	n	f	n	a	c	b	y	...
8121	e	f	s	n	f	n	a	c	b	n	...
8122	p	k	y	n	f	y	f	c	n	b	...
8123	e	x	s	n	f	n	a	c	b	y	...

8124 rows × 23 columns

In [35]:
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [38]:
```
df1['class'] = encoder.fit_transform(df1['class'])

df1['cap-shape'] = encoder.fit_transform(df1['cap-shape'])

df1['cap-surface'] = encoder.fit_transform(df1['cap-surface'])

df1['cap-color'] = encoder.fit_transform(df1['cap-color'])

df1['bruises'] = encoder.fit_transform(df1['bruises'])

df1['odor'] = encoder.fit_transform(df1['odor'])

df1['gill-attachment'] = encoder.fit_transform(df1['gill-attachment'])

df1['gill-spacing'] = encoder.fit_transform(df1['gill-spacing'])

df1['stalk-root'] = encoder.fit_transform(df1['gill-spacing'])

df1['gill-size'] = encoder.fit_transform(df1['gill-size'])

df1['gill-color'] = encoder.fit_transform(df1['gill-color'])

df1['stalk-shape'] = encoder.fit_transform(df1['stalk-shape'])

df1['stalk-surface-above-ring'] = encoder.fit_transform(df1['stalk-surface-above

df1['stalk-surface-below-ring'] = encoder.fit_transform(df1['stalk-surface-below
```

```
df1['stalk-color-above-ring'] = encoder.fit_transform(df1['stalk-color-above-rin

df1['stalk-color-below-ring'] = encoder.fit_transform(df1['stalk-color-below-rin

df1['veil-type'] = encoder.fit_transform(df1['veil-type'])

df1['veil-color'] = encoder.fit_transform(df1['veil-color'])

df1['ring-number'] = encoder.fit_transform(df1['ring-number'])

df1['ring-type'] = encoder.fit_transform(df1['ring-type'])

df1['spore-print-color'] = encoder.fit_transform(df1['spore-print-color'])

df1['population'] = encoder.fit_transform(df1['population'])

df1['habitat'] = encoder.fit_transform(df1['habitat'])
```

In [39]:
```
df1
```

Out[39]:

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...
0	1	5	2	4	1	6	1	0	1	4	...
1	0	5	2	9	1	0	1	0	0	4	...
2	0	0	2	8	1	3	1	0	0	5	...
3	1	5	3	8	1	6	1	0	1	5	...
4	0	5	2	3	0	5	1	1	0	4	...
...	...	...	...	...	...	...	...	...	...	...	...
8119	0	3	2	4	0	5	0	0	0	11	...
8120	0	5	2	4	0	5	0	0	0	11	...
8121	0	2	2	4	0	5	0	0	0	5	...
8122	1	3	3	4	0	8	1	0	1	0	...
8123	0	5	2	4	0	5	0	0	0	11	...

8124 rows × 23 columns

In [40]:
```
X = df1.iloc[:,1:24]
Y = df1.iloc[:,0:1]
```

In [102…
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random
```

In [103…
```
from sklearn.linear_model import LogisticRegression
dt=LogisticRegression()
```

In [104…
```
dt.fit(X_train,Y_train)
```

```
C:\Users\nayan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\utils\validation.py:1300: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples, ), for ex
ample using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\nayan\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn
\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (sta
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[104…
```
LogisticRegression()
```

In [105…
```
Y_pred = dt.predict(X_test)
Y_pred
```

Out[105…
```
array([0, 1, 0, ..., 0, 0, 0], dtype=int64)
```

In [106…
```
dt.score(X_test, Y_test)
```

Out[106…
```
0.9502708025603152
```

In [109…
```
from sklearn.metrics import accuracy_score,f1_score,recall_score,precision_score
```

In [110…
```
print('Accuracy: %.3f' % accuracy_score(Y_test, Y_pred))
print('f1 score: %.3f' % f1_score(Y_test, Y_pred,average='micro'))
print('recall: %.3f' % recall_score(Y_test, Y_pred, average='macro'))
print('Precision: %.3f' % precision_score(Y_test, Y_pred,average='macro'))
# Print confusion matrix
```

```
Accuracy: 0.950
f1 score: 0.950
recall: 0.950
Precision: 0.950
```

In [111…
```
df1.corr()
```

Out[111...

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment
class	1.000000	0.052951	0.178446	-0.031384	-0.501530	-0.093552	0.129200
cap-shape	0.052951	1.000000	-0.050454	-0.048203	-0.035374	-0.021935	0.078865
cap-surface	0.178446	-0.050454	1.000000	-0.019402	0.070228	0.045233	-0.034180
cap-color	-0.031384	-0.048203	-0.019402	1.000000	-0.000764	-0.387121	0.041436
bruises	-0.501530	-0.035374	0.070228	-0.000764	1.000000	-0.061825	0.137359
odor	-0.093552	-0.021935	0.045233	-0.387121	-0.061825	1.000000	-0.059590
gill-attachment	0.129200	0.078865	-0.034180	0.041436	0.137359	-0.059590	1.000000
gill-spacing	-0.348387	0.013196	-0.282306	0.144259	-0.299473	0.063936	0.071489
gill-size	0.540024	0.054050	0.208100	-0.169464	-0.369596	0.310495	0.108984
gill-color	-0.530566	-0.006039	-0.161017	0.084659	0.527120	-0.129213	-0.128567
stalk-shape	-0.102019	0.063794	-0.014123	-0.456496	0.099364	0.459766	0.186485
stalk-root	-0.348387	0.013196	-0.282306	0.144259	-0.299473	0.063936	0.071489
stalk-surface-above-ring	-0.334593	-0.030417	0.089090	-0.060837	0.460824	0.118617	-0.088916
stalk-surface-below-ring	-0.298801	-0.032591	0.107965	-0.047710	0.458983	0.061820	-0.116177
stalk-color-above-ring	-0.154003	-0.031659	0.066050	0.002364	0.083538	0.174532	0.099299
stalk-color-below-ring	-0.146730	-0.030390	0.068885	0.008057	0.092874	0.169407	0.097160
veil-type	NaN	NaN	NaN	NaN	NaN	NaN	NaN
veil-color	0.145142	0.072560	-0.016603	0.036130	0.119770	-0.057747	0.897518
ring-number	-0.214366	-0.106534	-0.026147	-0.005822	0.056788	0.111905	0.093236
ring-type	-0.411771	-0.025457	-0.106407	0.162513	0.692973	-0.281387	-0.146689
spore-print-color	0.171961	-0.073416	0.230364	-0.293523	-0.285008	0.469055	-0.029524
population	0.298686	0.063413	0.021555	-0.144770	0.088137	-0.043623	0.165575
habitat	0.217179	-0.042221	0.163887	0.033925	-0.075095	-0.026610	-0.030304

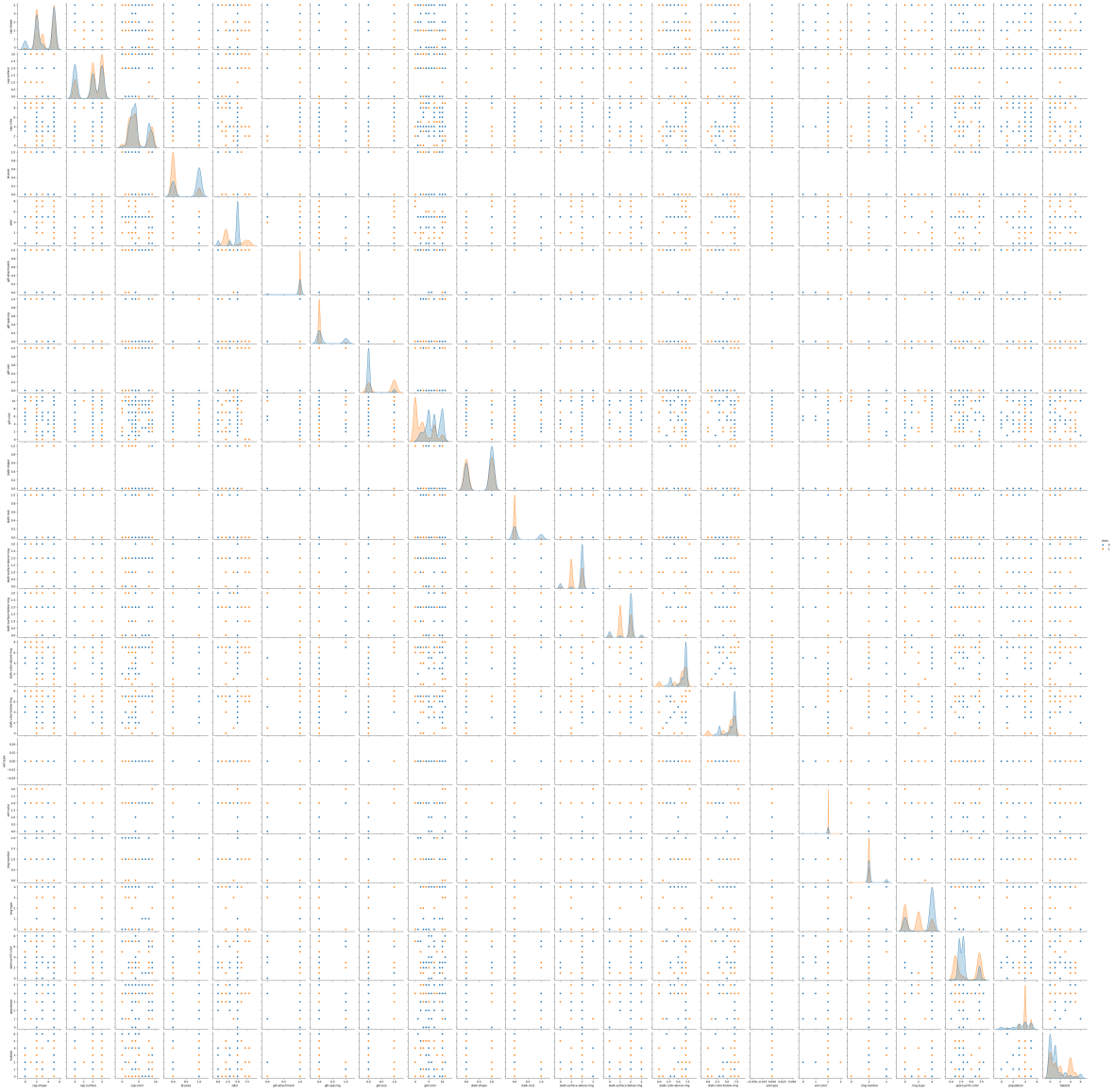
23 rows × 23 columns

In [112…

```
import seaborn as sns
sns.pairplot(df1, hue='class')
```

Out[112…

```
<seaborn.axisgrid.PairGrid at 0x228bce02650>
```



In [ ]: