In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

In [2]:
```
df = sns.load_dataset('iris')
```

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

Out[3]:

	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 [4]:
```
df['species'].value_counts() # we have 3 classes (multi class classification)
# but we foucus on binary classification
```

Out[4]:
```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [6]:
```
df = df.drop(df[df['species']=='setosa'].index,axis=0)
```

In [7]:
```
df.shape
```

Out[7]:
```
(100, 5)
```

In [8]:
```
df['species'].unique()
```

Out[8]:
```
array(['versicolor', 'virginica'], dtype=object)
```

In [9]:
```
df['species']=df['species'].map({'virginica':1,'versicolor':0})
```

In [10]:
```
df
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width	species
50	7.0	3.2	4.7	1.4	0
51	6.4	3.2	4.5	1.5	0
52	6.9	3.1	4.9	1.5	0
53	5.5	2.3	4.0	1.3	0
54	6.5	2.8	4.6	1.5	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	1
146	6.3	2.5	5.0	1.9	1
147	6.5	3.0	5.2	2.0	1
148	6.2	3.4	5.4	2.3	1
149	5.9	3.0	5.1	1.8	1

100 rows × 5 columns

In [11]:
```
##### spliting into independent and dependent features
```

In [12]:
```
x = df.iloc[:,:-1]
y=df.iloc[:,-1] #integer indexers are allowred
```

In [13]:
```
df.loc[:,['sepal_width']] #df.loc[rows,columns] # integer type indexers are not allowed
```

Out[13]:

	sepal_width
50	3.2
51	3.2
52	3.1
53	2.3
54	2.8
...	...
145	3.0
146	2.5
147	3.0
148	3.4
149	3.0

100 rows × 1 columns

In [14]:
```
x
```

Out[14]:

	sepal_length	sepal_width	petal_length	petal_width
50	7.0	3.2	4.7	1.4
51	6.4	3.2	4.5	1.5
52	6.9	3.1	4.9	1.5
53	5.5	2.3	4.0	1.3
54	6.5	2.8	4.6	1.5
...	...	...	...	...
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

100 rows × 4 columns

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=42)
```

In [17]:
```
from sklearn.linear_model import LogisticRegression
```

In [18]:
```
classifier = LogisticRegression()
```

In [19]:
```
from sklearn.model_selection import GridSearchCV
```

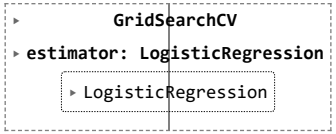
In [20]:
```
parameters = {'penalty':['l1','l2','elasticnet'],'C':[1,2,3,4,5,6,7,8,20,30,40],'max_iter':[100,200,300]}
```

In [21]:
```
classifier_regressor = GridSearchCV(classifier,param_grid=parameters,scoring='accuracy',cv=5) # accuray for classification problem
```

In [22]:
```
classifier_regressor.fit(x_train,y_train)
```

```
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan]
  warnings.warn(
```

Out[22]:



In [23]:
```
classifier_regressor.best_params_
```

Out[23]:
```
{'C': 1, 'max_iter': 100, 'penalty': 'l2'}
```

In [24]:
```
classifier_regressor.best_score_
```

Out[24]:
```
0.9733333333333334
```

In [25]:
```
##prediction
y_pred = classifier_regressor.predict(x_test)
```

In [26]:
```
from sklearn.metrics import accuracy_score,classification_report
```

In [27]:
```
accuracy_score(y_pred,y_test)
```

Out[27]:
```
0.92
```

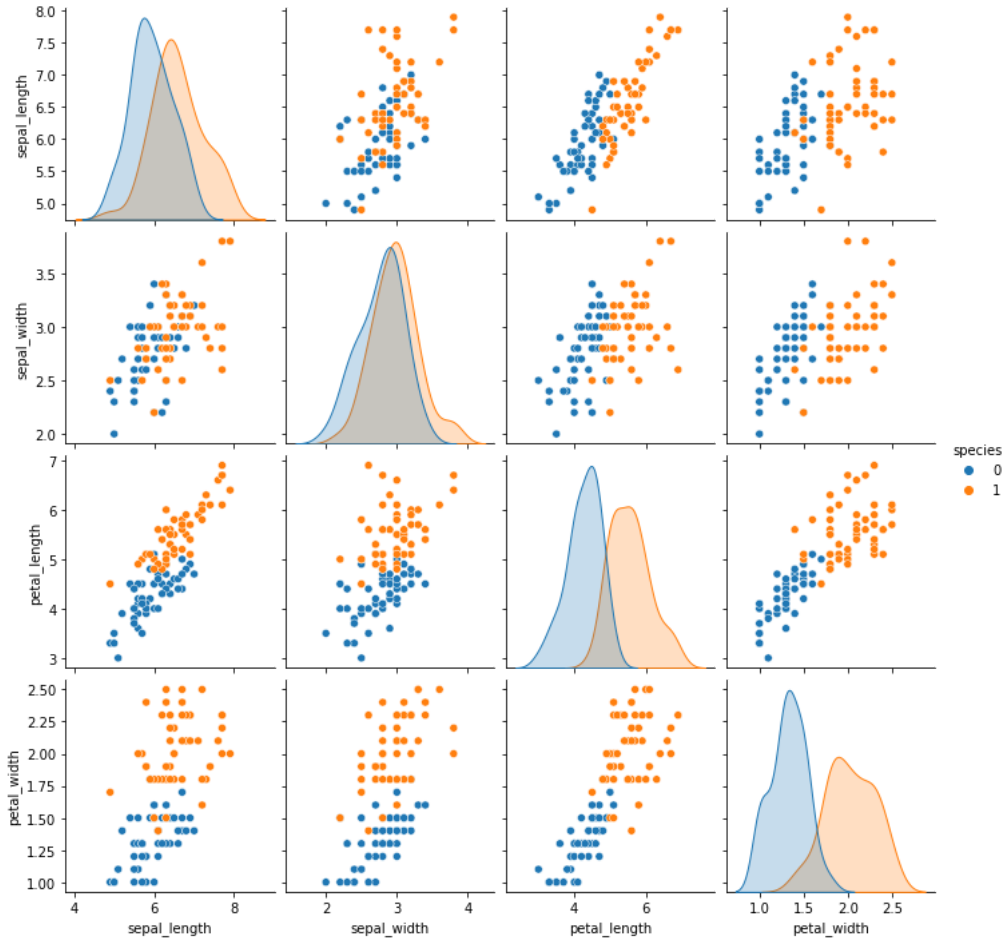
In [28]:
```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.91      0.91      0.91        11

    accuracy                           0.92        25
   macro avg       0.92      0.92      0.92        25
weighted avg       0.92      0.92      0.92        25
```

In [29]:
```
## EDA
```

In [30]:
```
sns.pairplot(df,hue='species')
```

Out[30]:
```
<seaborn.axisgrid.PairGrid at 0x1d38425ee90>
```



In [31]:
```
df.corr()
```

Out[31]:

	sepal_length	sepal_width	petal_length	petal_width	species
sepal_length	1.000000	0.553855	0.828479	0.593709	0.494305
sepal_width	0.553855	1.000000	0.519802	0.566203	0.308080
petal_length	0.828479	0.519802	1.000000	0.823348	0.786424
petal_width	0.593709	0.566203	0.823348	1.000000	0.828129
species	0.494305	0.308080	0.786424	0.828129	1.000000