

```
In [5]: import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

```
In [6]: df = sns.load_dataset('iris')
df.head()
```

```
Out[6]:
```

	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 [7]: df['species'].unique()
```

```
Out[7]: array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
In [8]: df.isnull()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [9]: `df.isnull().sum()`Out[9]: `sepal_length 0
sepal_width 0
petal_length 0
petal_width 0
species 0
dtype: int64`In [10]: `df['species'] != 'setosa'`

```
Out[10]: 0      False
         1      False
         2      False
         3      False
         4      False
         ...
        145     True
        146     True
        147     True
        148     True
        149     True
Name: species, Length: 150, dtype: bool
```

```
In [11]: df['species']==='setosa' # another way
```

```
Out[11]: 0      True
         1      True
         2      True
         3      True
         4      True
         ...
        145     False
        146     False
        147     False
        148     False
        149     False
Name: species, Length: 150, dtype: bool
```

```
In [12]: df = df[df['species']!='setosa'] # Here all things will be shown except setosa
```

```
In [13]: df.head()
```

```
Out[13]:
```

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

```
In [14]: # In Logistic Regression we cannot directly give versicolor and virginica, so we have to make a Numerical value.
```

```
In [15]: df['species'] = df['species'].map({'versicolor':0, 'virginica':1})
```

```
In [16]: df['species']
```

```
Out[16]: 50    0  
51    0  
52    0  
53    0  
54    0  
..  
145   1  
146   1  
147   1  
148   1  
149   1  
Name: species, Length: 100, dtype: int64
```

```
In [17]: df.head()
```

Out[17]:

	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

In [18]:

```
### Split Datasets into Dependent and Independent Features
X = df.iloc[:, :-1] # Take Every Column Except Last Column
y = df.iloc[:, -1]
```

In [19]:

```
X
```

Out[19]:

	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 [20]:

y

```
Out[20]: 50    0
      51    0
      52    0
      53    0
      54    0
      ..
     145   1
     146   1
     147   1
     148   1
     149   1
```

Name: species, Length: 100, dtype: int64

```
In [21]: from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
In [22]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state = 42)
```

```
In [23]: classifier = LogisticRegression()
```

```
In [24]: from sklearn.model_selection import GridSearchCV
```

```
In [25]: parameter = {'penalty':['l1','l2', 'elasticnet'], 'C': [1,2,3,4,5,6,10,20,30,40,50], 'max_iter':[100,200,300]}
```

```
In [47]: classifier_regressor = GridSearchCV(classifier, param_grid = {
    'solver': ['lbfgs'], # ya 'saga'
    'penalty': ['l2'], # lbfgs sirf l2 ya None accept karta hai
    'C': [0.1, 1, 10]
},
, scoring='accuracy', cv=5)
```

```
In [49]: classifier_regressor.fit(X_train, y_train)
```

Out[49]:

```

▶      GridSearchCV
          ⓘ ⓘ
        ▶      best_estimator_:
              LogisticRegression
                  ▶ LogisticRegression ⓘ

```

```
In [28]: print(classifier_regressor.best_params_)

{'C': 1, 'max_iter': 100, 'penalty': 'l2'}
```

```
In [29]: print(classifier_regressor.best_score_)

0.9733333333333334
```

```
In [30]: ## Prediction
y_pred = classifier_regressor.predict(X_test)
```

```
In [31]: y_pred
```

```
Out[31]: array([1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
```

```
    0, 0, 1])
```

```
In [32]: ## Accuracy Score
```

```
from sklearn.metrics import accuracy_score, classification_report
```

```
In [33]: score = accuracy_score(y_pred, y_test)
```

```
print(score)
```

```
0.92
```

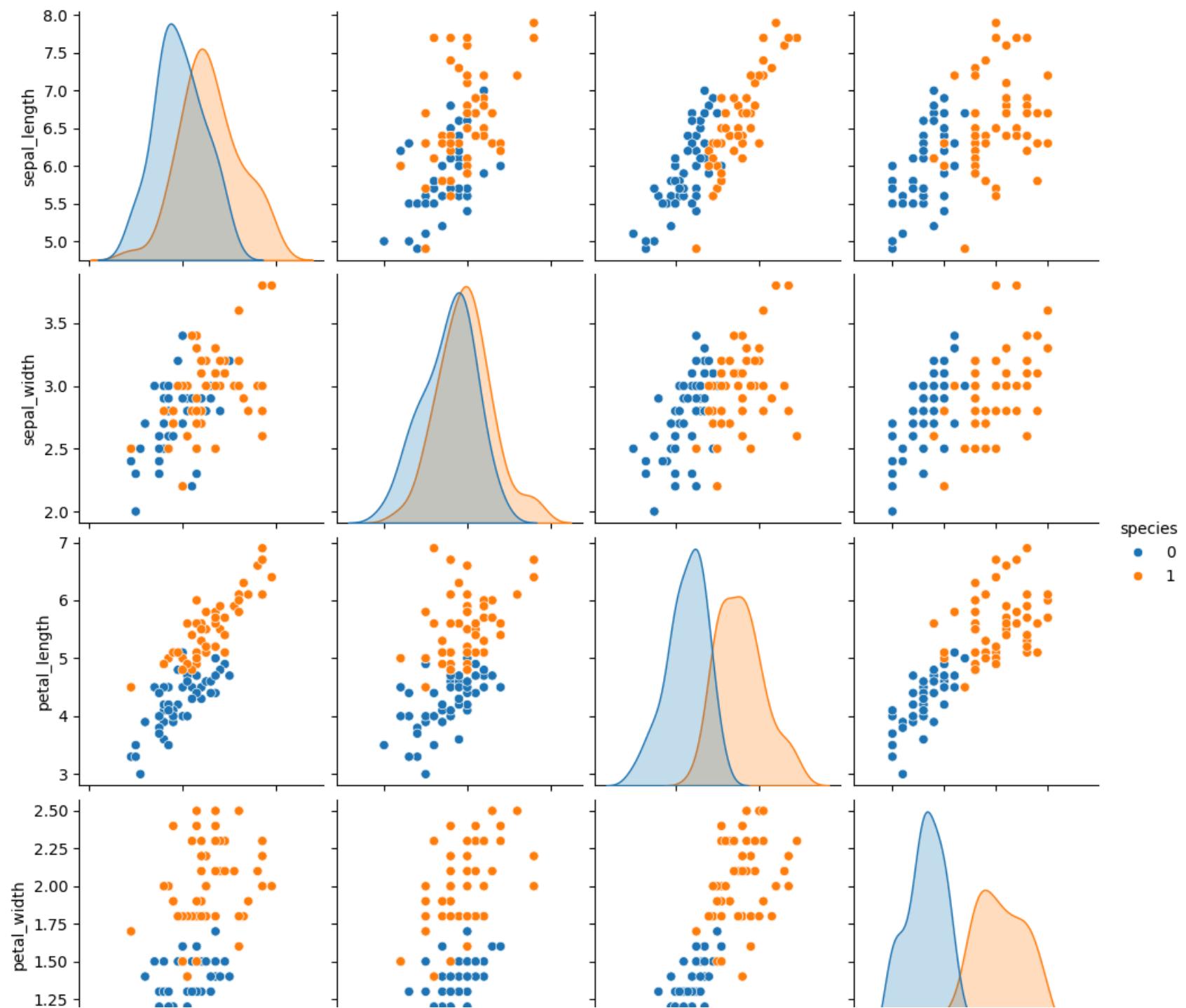
```
In [34]: print(classification_report(y_pred, y_test))
```

	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

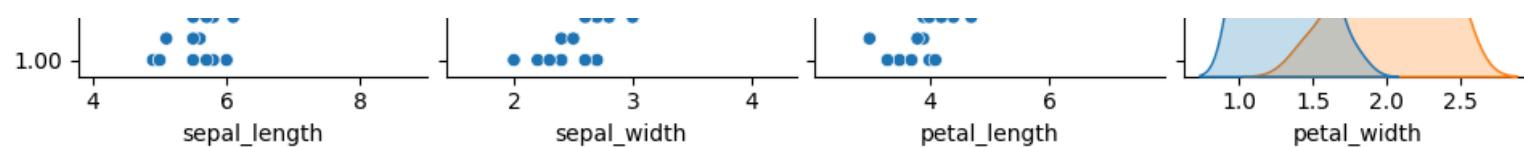
```
In [35]: ## EDA
```

```
sns.pairplot(df, hue='species')
```

```
Out[35]: <seaborn.axisgrid.PairGrid at 0x1655c50e3c0>
```



Logistic-Regression



```
In [36]: df.corr()
```

Out[36]:

	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