Logistic Regression

- Logistic Regression is a statistical model used for binary classification tasks. It predicts the probability that an input belongs to one of two possible classes, typically denoted as 0 and 1.
- Logistic Regression uses the sigmoid (logistic) function to map the linear combination of input features to a probability value between 0 and 1. P(Y=1|X) = 1 / (1 + e^(-(b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n)))

  = 1/ (1 + e^ -z)

  - The model minimizes a loss function (e.g., log loss) to find the best fit.

In [1]:
```
import seaborn as sns
import pandas as pd
import numpy as np
```

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

Out[2]:

	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 [3]:
```
df['species'].unique()
```

Out[3]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [5]:
```
# Only taking two classes
df = df[df['species']!='setosa']
```

In [6]:
```
df.head()
```

Out[6]:

	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 [7]:
```
df['species'] = df['species'].map({'versicolor':0, 'virginica':1})
```

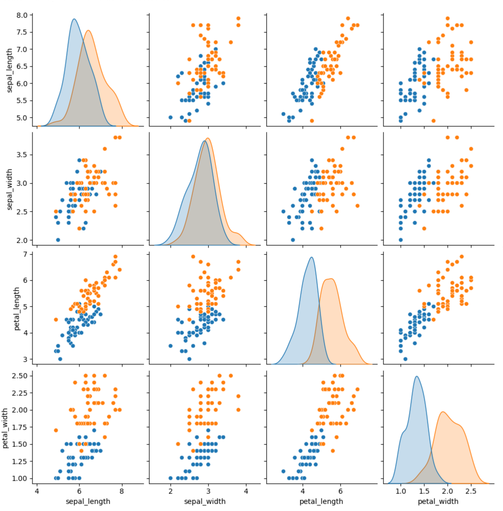
In [8]:
```
df.head()
```

Out[8]:

	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 [9]:
```
sns.pairplot(df, hue = 'species')
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1e0d9b084c0>
```



In [10]:
```
df.corr()
```

Out[10]:

	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

In [11]:
```
# Split dataset into independent and dependent features
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [12]:
```
X
```

Out[12]:

	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 [13]:
```
y
```

Out[13]:
```
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 [14]:
```
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_state = 42)
```

In [15]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
```

In [16]:
```
from sklearn.model_selection import GridSearchCV
parameter = {'penalty': ['l1','l2','elasticnet'], 'C':[1,2,3,4,5,6,10,20,30,40,50], 'max_iter':[100,200,300]}
```

In [17]:
```
classifier_regressor = GridSearchCV(classifier, param_grid = parameter, scoring = 'accuracy', cv = 5 )
```

In [18]:
```
classifier_regressor.fit(X_train, y_train)
```

```
C:\Users\Garima\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:425: FitFailedWarning:
330 fits failed out of a total of 495.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
165 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\base.py", line 1151, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1168, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 56, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

--------------------------------------------------------------------------------
165 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\base.py", line 1151, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 1168, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "C:\Users\Garima\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py", line 56, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got elasticnet penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\Garima\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:976: UserWarning: One or more of the test scores are non-finite: [       nan 0.97333333        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
        nan 0.97333333        nan        nan 0.97333333        nan
        nan 0.97333333        nan]
  warnings.warn(
```

Out[18]:

GridSearchCV
estimator: LogisticRegression
LogisticRegression

In [19]:
```
print(classifier_regressor.best_params_)
```

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

In [20]:
```
print(classifier_regressor.best_score_)
```

```
0.9733333333333334
```

In [21]:
```
#Prediction
y_pred = classifier_regressor.predict(X_test)
```

In [22]:
```
# Accuracy_Score
from sklearn.metrics import accuracy_score, classification_report
```

In [23]:
```
score = accuracy_score(y_pred, y_test)
print(score)
```

```
0.92
```

In [24]:
```
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 [ ]:
```
```