1.1 Introduction to Logistic Regression

- Logistic Regression is a statistical method used for binary classification problems.
- Unlike Linear Regression, which predicts a continuous value, Logistic Regression predicts the probability that a given input belongs to a particular class.
- In simple terms, it outputs probabilities between 0 and 1, which can be interpreted as a classification between two classes (e.g., success/failure, yes/no).

- Mathematical Representation: Logistic Regression uses the sigmoid function to map predicted values (logits) to probabilities.

The sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$

Where $z = X \cdot \theta$, and θ represents the parameters (weights) of the model.

In [5]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Example dataset (Titanic dataset)

df = sns.load_dataset('titanic')

# Select relevant features and the target variable
X = df[['pclass', 'age', 'fare', 'sibsp', 'parch']].fillna(0)
y = df['survived']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=1)

# Create and train logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.65
```

1.2 Interpretation from a Logistic Regression model

- In Logistic Regression, we are interested in understanding how changes in the input variables (features) affect the probability of the target outcome.
- The coefficients of the model (also called weights) provide insight into the relationship between features and the target.
- For binary classification:
  - A positive coefficient increases the likelihood of the outcome.

- A negative coefficient decreases the likelihood.

In [6]:

```
# Coefficients of the model
coefficients = pd.DataFrame({
    'Feature': X_train.columns,
    'Coefficient': model.coef_[0]
})
print(coefficients)
```

```
  Feature  Coefficient
0  pclass    -0.934142
1     age    -0.016394
2    fare     0.003250
3   sibsp    -0.199094
4   parch     0.378752
```

NOTE- Interpret the sign and magnitude of each coefficient to understand its influence on the target.

1.3 Changing the threshold of a Logistic Regression model

- By default, Logistic Regression uses a threshold of 0.5 to classify observations (i.e., probabilities ≥ 0.5 are classified as class 1, and < 0.5 as class 0).
- However, we can modify this threshold to favor precision, recall, or other evaluation metrics depending on the problem.

In [7]:

```
# Predict probabilities
y_prob = model.predict_proba(X_test)[:, 1]

# Change threshold to 0.7 instead of 0.5
threshold = 0.7
y_pred_threshold = np.where(y_prob >= threshold, 1, 0)

# Evaluate accuracy with new threshold
accuracy_threshold = accuracy_score(y_test, y_pred_threshold)
print(f"Accuracy with threshold 0.7: {accuracy_threshold:.2f}")
```

```
Accuracy with threshold 0.7: 0.63
```

NOTE- You can adjust the threshold depending on the trade-off between false positives and false negatives in your classification task.

1.4 Evaluation of a Classification Model

- To evaluate the performance of a logistic regression model, we use several metrics, including:
  - Accuracy: The proportion of correct predictions.
  - Precision: The proportion of positive predictions that are actually correct.
  - Recall: The proportion of actual positives that were identified correctly.
  - F1-Score: The harmonic mean of precision and recall.
  - Confusion Matrix: A matrix showing true positives, false positives, true negatives, and false negatives.

In [8]:

```python
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix

# Precision
precision = precision_score(y_test, y_pred)
# Recall
recall = recall_score(y_test, y_pred)
# F1-Score
f1 = f1_score(y_test, y_pred)
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)

print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
print("Confusion Matrix:\n", conf_matrix)
```

```
Precision: 0.60
Recall: 0.42
F1-Score: 0.50
Confusion Matrix:
 [[85 21]
 [42 31]]
```

1.4 Pros and Cons of Logistic Regression

- Pros:
  - Simple and interpretable model.
  - Works well for binary classification problems.
  - Outputs probabilities, which can be useful for decision-making.
  - Can handle non-linear boundaries by using polynomial or interaction terms.
- Cons:
  - Assumes a linear relationship between the independent variables and the log odds.
  - It may not perform well with highly complex datasets unless additional transformations are applied.
  - Overfitting is possible, especially if there are too many irrelevant features.

****