LOGISTIC REGRESSION

BY RAJENDRA PHANI

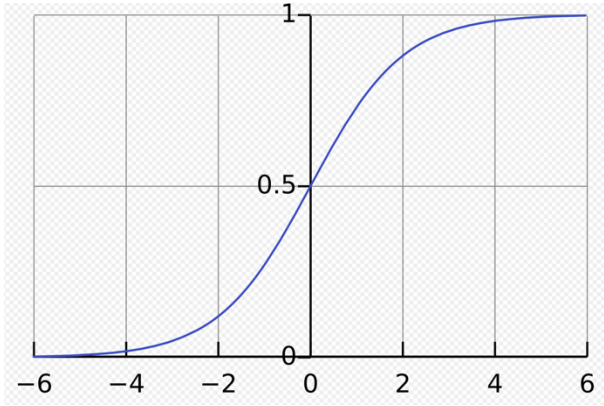
- Logistic regression is a supervised machine learning algorithm mainly used for classification problems, especially when the output to predict is a binary variable (like yes/no, 0/1, true/false). Instead of predicting a continuous number like linear regression, logistic regression predicts the probability that a given input belongs to a particular class.

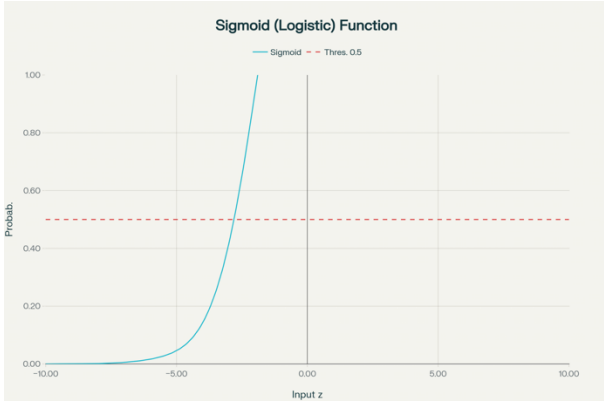
- **Intuition:**
- Imagine you want to predict whether a student passes an exam (yes/no) based on their number of study hours.
- Instead of predicting marks directly, logistic regression estimates the chance of passing, giving a value between 0 and 1. It uses a special curve called the sigmoid function (or logistic function), which produces an S-shaped curve. This curve "squeezes" any input from minus infinity to plus infinity into the range 0 to 1, making it suitable to represent probabilities.
- **How it works:**
- It finds the best linear combination of input features (like study hours, attendance, etc.) and passes this through the sigmoid function. If the result is close to 1, it predicts "yes" (pass); if close to 0, it predicts "no" (fail).
- You can set a threshold (commonly 0.5): if the probability is greater than 0.5, predict "yes"; otherwise, predict "no".
- **Key points:**
- Used for binary classification (two classes), but can be extended to more classes (multinomial or ordinal logistic regression).
- The output is always a probability, making the predictions interpretable.
- Widely used due to its simplicity, interpretability, and efficiency in many fields like medicine, finance, and marketing.

Advantages	Disadvantages
Simple to implement, interpret, and train.	Assumes a linear relationship between input features and the log-odds, which may not always hold.
Outputs well-calibrated probabilities for binary classification tasks.	Limited to linear decision boundaries, making it ineffective for complex relationships.
Computationally efficient and fast to converge, even on large datasets.	Sensitive to irrelevant or highly correlated input features (multicollinearity).
Works well when the relationship between feature variables and outcome is almost linear.	Struggles when there are more features than samples, or with noisy data.
Provides insight into the importance and impact of each predictor.	May be less accurate than more complex models (e.g., tree-based or ensemble methods).
Can be updated easily with new data (incremental learning).	Requires careful feature engineering and standardization for optimal performance.

- Formula of Logistic Regression
- The logistic regression model predicts the probability that an input belongs to the positive class (usually denoted as 1). The core formula is:

- where:
- • is the sigmoid function,
- • is a weighted sum of the input features,
- • is the intercept (bias),
- • are the feature weights,
- • are feature values.
- The output is always between 0 and 1, representing a probability.





$$P(y=1|x) = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

If $\sigma(z) > 0.5$, predict class 1.

If $\sigma(z) \leq 0.5$, predict class 0.

When z is very large and positive, $\sigma(z)$ approaches 1.

When z is very small and negative, $\sigma(z)$ approaches 0.

When $z = 0$, $\sigma(0) = 0.5$.

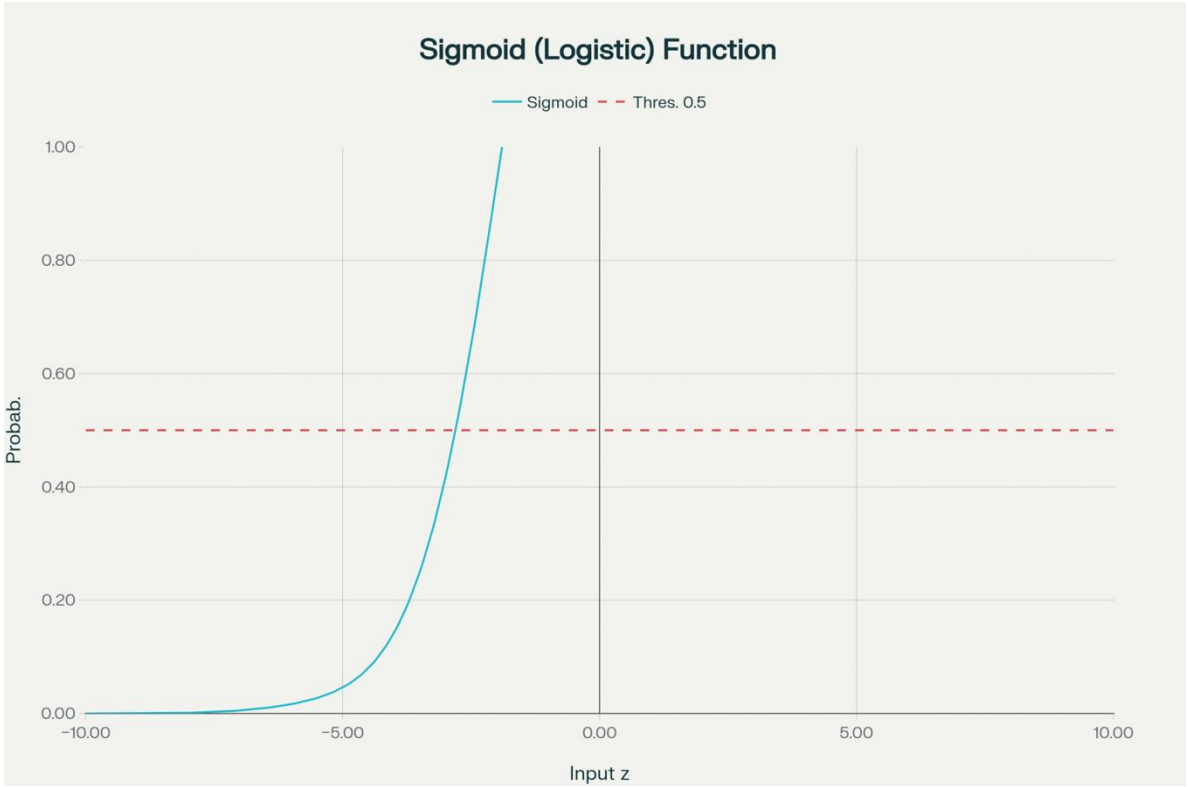
$$z = w_0 + w_1 x_1 + w_2 x_2 + + w_n x_n$$

- **Detailed Explanation**
  - First, logistic regression computes a linear combination of features

  - This value can be any real number from negative to positive infinity.

- **Sigmoid Function:**
- To convert into a probability, we use the sigmoid (logistic) function:

- **Prediction and Classification :**
- The output of the sigmoid function is compared to a threshold (often 0.5) to make a decision:



- **Key Insights:**
- Intuitive interpretation: Logistic regression models the odds and their logarithm (log-odds, or logit) as a linear function of input variables.
- Why sigmoid?: The sigmoid ensures outputs remain in the interval, matching probability requirements.
- Threshold use: By adjusting the threshold, you can tune the sensitivity of your classifier to false positives or false negatives.

$$z = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

- $x_1, x_2, , x_n$ are input feature values.
- $w_0, w_1, , w_n$ are the model's weights (parameters).

- **Math Behind Logistic Regression**
- Logistic regression is a classification algorithm used to predict the probability of a binary outcome based on input features. It's rooted in statistics and probability theory, especially the concepts of log-odds, the sigmoid function, and maximum likelihood estimation.
- Mathematical Foundations
- Linear Model
- Logistic regression starts with a linear combination of input features:
- The Sigmoid (Logistic) Function
- Since a linear combination can take any value from to , we need a function that maps this to a probability (0 to 1). The sigmoid does exactly that:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- If z is large and positive, $\sigma(z) \approx 1$.
- If z is large and negative, $\sigma(z) \approx 0$.
- At $z = 0$, $\sigma(z) = 0.5$.

Logistic Regression Probability

The predicted probability of the positive class:

$$P(y = 1|x) = \sigma(z) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + + w_n x_n)}}$$

Log-Odds (Logit Transformation)

Logistic regression models the **log-odds** (logit) as a linear function:

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = w_0 + w_1 x_1 + + w_n x_n$$

3. Loss Function and Model Training

Binary Cross-Entropy (Log-Loss)

The parameters (w) are learned by **maximizing the likelihood** of observed data (Maximum Likelihood Estimation), or equivalently, by minimizing the loss:

$$L = -\frac{1}{m}\sum_{i=1}^{m} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

- y_i: true binary label (0 or 1)

- Let's use: $w_0 = -4$, $w_1 = 2$.

For a student who studies $x = 3$ hours:

$$z = -4 + 2 \times 3 = 2$$

$$\sigma(2) = \frac{1}{1 + e^{-2}} \approx 0.88$$

So, the model predicts an **88% chance to pass**.

5. Graphs

Sigmoid Function

The sigmoid produces an S-shaped curve:

z	-6	-2	0	2	6
σ(z)	0.002	0.12	0.50	0.88	0.998

If you plot z on the x-axis and $\sigma(z)$ on the y-axis, the curve starts near zero for large negative z, rises steeply arou
and flattens near one for large positive z.

6. Interpretation and Intuition

- **Coefficients (w)**: The sign and size of each weight tell you how strongly and in which direction each input fea
influences the log-odds, and thus, the probability.
- **Threshold**: Typically, 0.5 is the cutoff: if $P(y = 1|x) > 0.5$, predict "1"; otherwise, "0".
- **Extension**: Can be generalized for multi-class classification (multinomial logistic regression).

- Example
- Suppose we want to predict whether a student passes (1) or fails (0) based on hours studied ().

- **Key Points:**

- Logistic regression is not about fitting a line, but modeling the probability of belonging to a class.

- The math hinges on transforming a linear prediction through the sigmoid function to obtain good probabilistic outputs.

- Training optimizes weights to minimize the error in predicted probabilities for all training samples.
- This mathematical structure allows logistic regression to be both interpretable and powerful for many real-world classification tasks.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

- **TP:** Correctly predicted positive cases.
- **TN:** Correctly predicted negative cases.
- **FP:** Incorrectly predicted positive (Type I error).
- **FN:** Incorrectly predicted negative (Type II error) [1].

- Evaluation Metrics for Logistic Regression
- Evaluating a logistic regression model requires understanding a range of metrics. Each metric tells you something different about your model's performance, especially on classification tasks. Below are the most important metrics, their calculations, and practical examples.
- 1. **Confusion Matrix**
- A confusion matrix is a table summarizing the performance of a classification model:

Example:

Suppose you have a binary classification problem (spam vs. not spam):

	Predicted Spam	Predicted Not Spam
Actual Spam	80	20
Actual Not Spam	10	90

- TP = 80, FN = 20, FP = 10, TN = 90

2. Accuracy

Definition: Proportion of correct predictions among all predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Example Calculation:

$$\text{Accuracy} = \frac{80 + 90}{80 + 20 + 10 + 90} = \frac{170}{200} = 0.85$$

So, the model is 85% accurate [2] [1].

3. Precision

Definition: Out of all predicted positives, how many are actually positive?

$$\text{Precision} = \frac{TP}{TP + FP}$$

Calculation:

$$\text{Precision} = \frac{80}{80 + 10} = \frac{80}{90} \approx 0.89$$

4. Recall (Sensitivity/True Positive Rate)

Definition: Out of all actual positives, how many did the model correctly detect?

$$\text{Recall} = \frac{TP}{TP + FN}$$

Calculation:

$$\text{Recall} = \frac{80}{80 + 20} = \frac{80}{100} = 0.8$$

5. F1 Score

Definition: Harmonic mean of Precision and Recall.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Calculation:

$$F1 = 2 \times \frac{0.89 \times 0.8}{0.89 + 0.8} \approx 2 \times \frac{0.712}{1.69} \approx 0.84$$

6. Specificity (True Negative Rate)

Definition: Out of all actual negatives, how many did the model correctly identify?

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Calculation:

$$\text{Specificity} = \frac{90}{90 + 10} = \frac{90}{100} = 0.9$$

7. ROC Curve and AUC

- **ROC Curve:** Plots the True Positive Rate (Recall) against the False Positive Rate (FPR) at various thresholds.
- **AUC (Area Under the Curve):** Measures the overall ability of the model to discriminate between the two classes. Higher is better (max 1.0, chance is 0.5).

Summary Table of Metric Calculations

Metric	Formula	Example Value
Accuracy	$(TP+TN)/(TP+TN+FP+FN$	0.85
Precision	$TP/(TP+FP)$	0.89
Recall	$TP/(TP+FN)$	0.80
F1 Score	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	0.84
Specificity	$TN/(TN+FP)$	0.90
AUC	Area under ROC curve	0.92
Log-Loss	(see above)	-

Notes on Choosing Metrics

- For **balanced classes**, accuracy is reliable.

Logistic Regression Analysis : File Insurance Data

Step:1

Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
```

Step:2

2. Load Data

```
# Replace 'insurance.csv' with your file path
data = pd.read_csv('insurance.csv')
```

Step:3

3. Explore Data

```
print("First 5 rows:")
print(data.head())

print("\nData info (dtypes, non-null counts):")
print(data.info())

print("\nSummary statistics:")
print(data.describe())

print("\nCheck missing values:")
print(data.isnull().sum())
```

```
 4. Handle Missing Values (if any)
# For example, fill missing numeric with mean or drop rows if few missing
# Here is a generic way:
if data.isnull().sum().sum() > 0:
    data = data.dropna()  # or use data.fillna(data.mean()) depending on context

# 5. Feature Engineering / Encoding (if needed)
# If there are categorical variables, encode them (OneHotEncoding or LabelEncoding)
# Example:
categorical_cols = data.select_dtypes(include=['object']).columns
print("\nCategorical columns:", categorical_cols)

# Encoding categorical variables
data = pd.get_dummies(data, drop_first=True)

# 6. Define Features (X) and Target (y)
# Assume target column is named 'target' or 'response' or similar, replace with actual
target_column = 'target'  # <-- replace with your actual target column
X = data.drop(columns=[target_column])
y = data[target_column]
```

```
# 7. Split Data into Training and Testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 8. Initialize Logistic Regression Model and Train
model = LogisticRegression(max_iter=1000)  # Increasing max_iter to ensure convergence
model.fit(X_train, y_train)

# 9. Make Predictions
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

# 10. Evaluate Model Performance
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:\n", cm)

# Classification Report (Precision, Recall, f1-score)
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# ROC Curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
auc_score = roc_auc_score(y_test, y_proba)
print(f"\nROC AUC Score: {auc_score:.3f}")
```

11. Plotting

```
# Plot Confusion Matrix Heatmap
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.show()

# Plot ROC Curve
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.3f})')
plt.plot([0,1], [0,1], linestyle='--', color='r')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```