Robotics & Internet-of-Things

جامعة الأمير سلطان
PRINCE SULTAN UNIVERSITY

CS316: INTRODUCTION TO AI AND DATA SCIENCE

CHAPTER 7
CLASSIFICATION

LECTURE 1
INTRODUCTION TO CLASSIFICATION

Prof. Anis Koubaa

May 2024

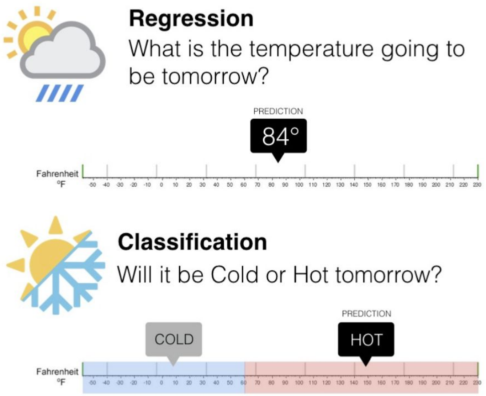
www.riotu-lab.org

Learning Outcomes

1. **Explain Classification Concepts:**
   - Differentiate between binary and multi-class classification.
   - Identify and describe accuracy, precision, recall, and F1-score.
2. **Apply Logistic Regression:**
   - Understand and implement logistic regression, interpreting its parameters.
3. **Interpret a Classification Report:**
   - Analyze classification reports to evaluate model accuracy and effectiveness.
4. **Use ROC for Model Evaluation:**
   - Construct and interpret ROC curves and calculate the Area Under the Curve (AUC) to assess model performance.

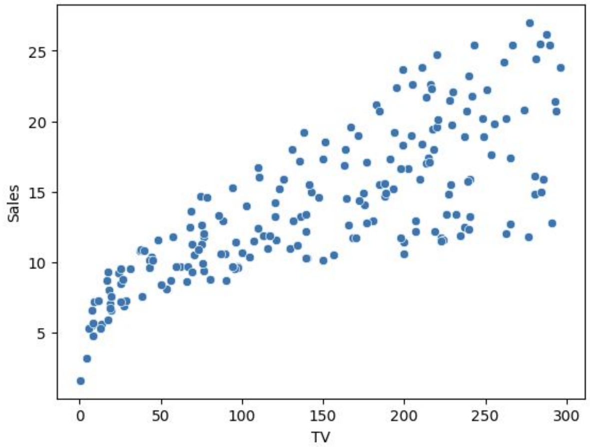
Introduction to Classification

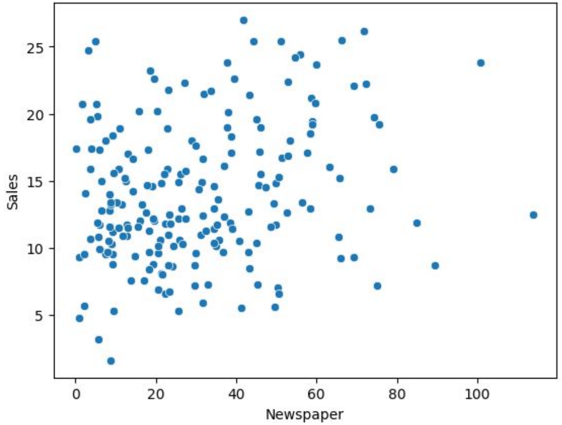
- **Definition:**
  - **Classification:** Assign labels from a finite set based on input features.
- **Mathematical Framework:**
  - Let $X \subseteq \mathbb{R}^n$ be the feature space and $Y = \{y_1, y_2, \ldots, y_k\}$ the label set.
  - Seek function $f : X \to Y$.
- **Comparison with Regression:**
  - **Regression:** $\hat{y} = f(x), f : X \to \mathbb{R}$ (Continuous output).
  - **Classification:** $\hat{y} = f(x), f : X \to Y$ (Discrete output).

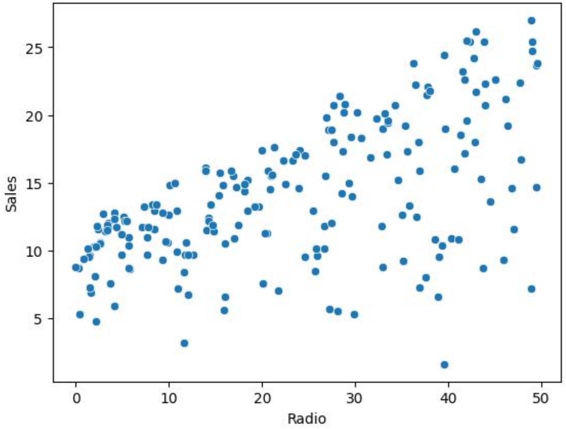


Regression Use Case

	ID	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9







Classification – Heart Disease Prediction

- **Features Explained:**
  - **Age:** Patient's age in years.
  - **Sex:** Patient's gender (1 = male, 0 = female).
  - **Cp (Chest Pain Type):** Type of chest pain experienced (values from 1 to 4).
  - **Trestbps (Resting Blood Pressure):** Resting blood pressure in mm Hg.
  - **Chol (Serum Cholesterol):** Serum cholesterol in mg/dl.
  - **Fbs (Fasting Blood Sugar):** Fasting blood sugar > 120 mg/dl (1 = true, 0 = false).
  - **Restecg (Resting ECG Results):** Results of electrocardiogram at rest (0, 1, or 2).
  - **Thalach (Max Heart Rate Achieved):** Maximum heart rate achieved.
  - **Exang (Exercise Induced Angina):** Angina induced by exercise (1 = yes, 0 = no).
  - **Oldpeak:** ST depression induced by exercise relative to rest.
  - **Slope:** Slope of the peak exercise ST segment.
  - **Ca:** Number of major vessels colored by fluoroscopy.
  - **Thal:** Thalassemia (3 = normal; 6 = fixed defect; 7 = reversible defect).
- **Target Variable (Highlighted):**
  - **Num (Diagnosis of Heart Disease):** The degree of heart disease (0 = no presence, 1-4 indicate varying degrees of heart disease).

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num	Target
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0	NO
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2	YES
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1	YES
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0	NO
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0	NO

CHAPTER 3
CLASSIFICATION

CLASSIFICATION

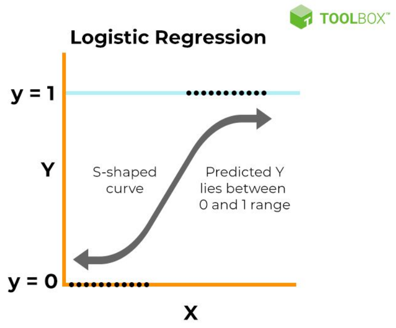
CLASSIFICATION TECHNIQUES

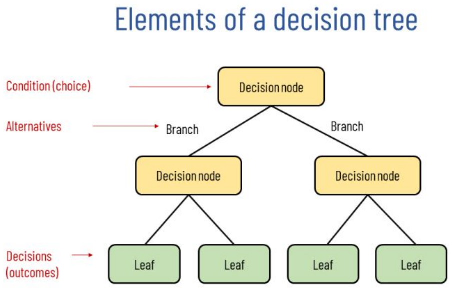
CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

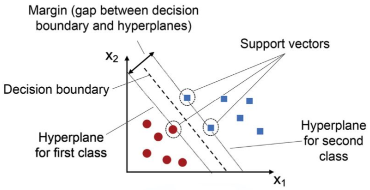
Major Classification Techniques

- **Purpose of Classification:**
  - To categorize data into predefined labels based on their attributes, aiding in predictive analysis across diverse applications.
- **Major Classification Techniques:**
  - **Logistic Regression:**
    - Suitable for binary classification tasks. Models the probability of the default class via the logistic function.
    - **Equation:** $\mathrm{logit}(p) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$
  - **Decision Trees:**
    - Splits the data into branches to make decisions. Useful for both categorical and continuous input and output variables.
    - **Criteria:** Information Gain, Gini Impurity.

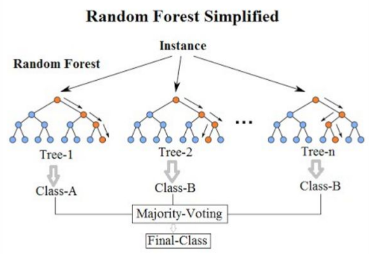


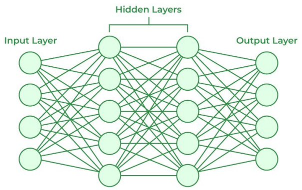


Major Classification Techniques

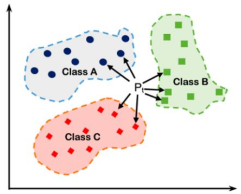


- **Support Vector Machines (SVM):**
  - Finds the hyperplane that best divides a dataset into classes with the maximum margin.
  - **Equation:** $\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \forall i$
- **Random Forests:**
  - An ensemble of decision trees designed to improve classification accuracy and control over-fitting.
  - **Mechanism:** Each tree votes for a class, and the majority vote decides the final class.
- **Neural Networks:**
  - Mimics the workings of the human brain to classify data based on deep learning techniques.
  - **Layers:** Input, hidden, and output layers with activation functions like Sigmoid, ReLU.
- **k-Nearest Neighbors (k-NN):**
  - Classifies new cases based on a similarity measure (e.g., distance functions).
  - **Rule:** Assign the class most common among the nearest k neighbors.





K Nearest Neighbors



CHAPTER 3
CLASSIFICATION

CLASSIFICATION

OUTPUT ENCODING

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Encoding the OUTPUT in Classification

- **Purpose of Encoding:**
  - Convert categorical variables into a numerical format suitable for algorithmic processing, enabling integration into mathematical models.
- **Label Encoding:**
  - **Definition:** Map each unique category to a unique integer.
  - **Example for 'Target':** Map 'NO' → 0 and 'YES' → 1.
  - **Mathematical Representation:** $\text{Target}_{\text{encoded}} = \begin{cases} 0 & \text{if Target} = \text{'NO'} \\ 1 & \text{if Target} = \text{'YES'} \end{cases}$

slope	ca	thal	num	Target
3	0.0	6.0	0	NO
2	3.0	3.0	2	YES
2	2.0	7.0	1	YES
3	0.0	3.0	0	NO
1	0.0	3.0	0	NO

Encoding the Target Variable in Python

- **Purpose of Label Encoding:**
  - Convert categorical text data into a model-readable numerical format.
- **Using Pandas for Label Encoding:**
  - **Code Example:**

python · Copy code

```python
import pandas as pd
data = {'Target': ['NO', 'YES', 'NO', 'YES']}
df = pd.DataFrame(data)
df['Target_encoded'] = df['Target'].astype('category').cat.codes
print(df)
```

```
  Target  Target_encoded
0     NO               0
1    YES               1
2     NO               0
3    YES               1
```

- **Key Points:**
  - Label encoding is straightforward but introduces a numerical order that may not exist, influencing some model types.

One-Hot Encoding

- **One Hot Encoding:**
  - **Definition:** Convert each category value into a binary vector representing the presence (1) or absence (0) of the feature.
  - **Mathematical Representation:** Define indicator function $\mathbf{1}_{\text{category}}(x)$ for each category in the variable.
  - **Example for 'Target':**
    - 'NO' → [1, 0]
    - 'YES' → [0, 1]
  - **Vector Form:**
    - $\text{Target}_{\text{NO}} = \mathbf{1}_{\text{'NO'}}(\text{Target})$
    - $\text{Target}_{\text{YES}} = \mathbf{1}_{\text{'YES'}}(\text{Target})$

slope	ca	thal	num	Target
3	0.0	6.0	0	NO
2	3.0	3.0	2	YES
2	2.0	7.0	1	YES
3	0.0	3.0	0	NO
1	0.0	3.0	0	NO

One-Hot Encoding - Scikit-learn

- **Purpose of One-Hot Encoding:**
  - Converts each category into a distinct binary column, which is crucial for models that do not assume any ordinal relationship between categories.
- **Updated Code for Scikit-learn One-Hot Encoding:**
  - Here's how to handle the one-hot encoding correctly using the latest Scikit-learn methods:

python Copy code

```python
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse_output=False)  # Corrected parameter usage
encoded = encoder.fit_transform(df[['Target']])  # Assuming df is predefined
df_encoded = pd.DataFrame(encoded, columns=encoder.get_feature_names_out(['Target
df = pd.concat([df, df_encoded], axis=1)
print(df)
```

- **Key Updates and Considerations:**
  - The use of `get_feature_names_out()` to retrieve column names from the encoder, replacing deprecated `get_feature_names()`.
  - Specifying `sparse_output=False` to handle the deprecation notice about the `sparse` parameter.

One-Hot Encoding with Scikit-learn

	Target	Target_NO	Target_YES
0	NO	1.0	0.0
1	YES	0.0	1.0
2	NO	1.0	0.0
3	YES	0.0	1.0

One-Hot Encoding - Pandas

- **Alternative Method with Pandas:**
  - Often simpler and more direct for handling within a DataFrame context.
- **Using Pandas for One-Hot Encoding:**
  - **Code Example:**

```python
df_one_hot = pd.get_dummies(df['Target'], prefix='Target')
df = pd.concat([df, df_one_hot], axis=1)
print(df)
```

- **Considerations:**
  - `get_dummies` is very efficient and directly integrates with pandas DataFrames, making it ideal for exploratory data analysis and preliminary data processing.

One-Hot Encoding with Pandas

	Target	Target_NO	Target_YES	Target_NO	Target_YES
0	NO	1.0	0.0	True	False
1	YES	0.0	1.0	False	True
2	NO	1.0	0.0	True	False
3	YES	0.0	1.0	False	True

CHAPTER 3
CLASSIFICATION

CLASSIFICATION

Logistic Function (Sigmoid)

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

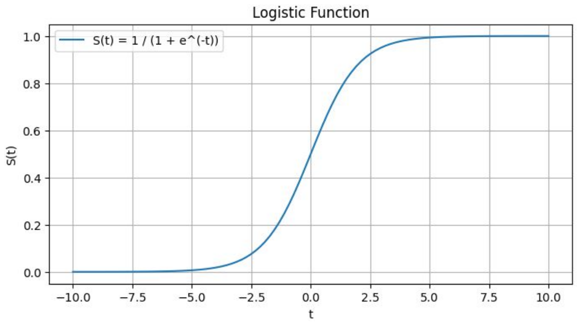
LOGISTIC REGRESSION

- **Definition:**
  - Logistic Regression is a statistical method for binary classification that models the probability of a binary response based on one or more predictor variables.
- **Mathematical Representation:**
  - It estimates probabilities using the logistic function, which is an S-shaped curve that maps any real-valued number into the (0, 1) interval, making it suitable for modeling probability.
- **Logistic Function:**
  - The logistic function is defined as:

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

  - Where z is the linear combination of the input features X and their corresponding coefficients β, expressed as:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$



- **Purpose:**
  - **Predictive Modelling:** Predicts the probability P that an observation falls into the category $Y = 1$ (positive class), as opposed to $Y = 0$ (negative class).
  - **Decision Boundary:** The decision boundary in logistic regression, which is set at $\sigma(z) = 0.5$, determines the threshold at which the probability output is converted into class labels.
- **Mathematical Goal:**
  - **Maximize Likelihood:** The parameters β are typically estimated using maximum likelihood estimation (MLE), which seeks to maximize the likelihood function, ensuring the best model coefficients that predict the observed outcomes.

Logit in Logistic Regression

- **Logit to Sigmoid Relationship:**
  - The logistic regression model begins with a linear combination of input features x_i and their respective coefficients β_i:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

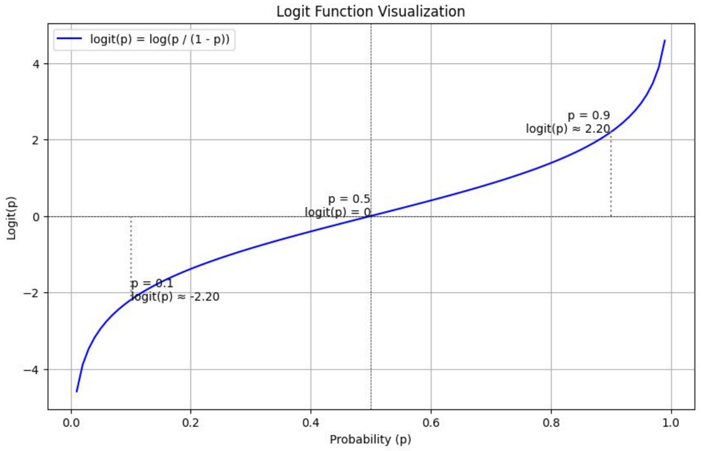
  - The linear combination z then becomes the input to the logit function, which computes the log-odds of a positive outcome:

$$\text{logit}(P(Y = 1|X)) = \log\left(\frac{P(Y = 1|X)}{1 - P(Y = 1|X)}\right) = z$$

  - This relationship can be expressed more explicitly:

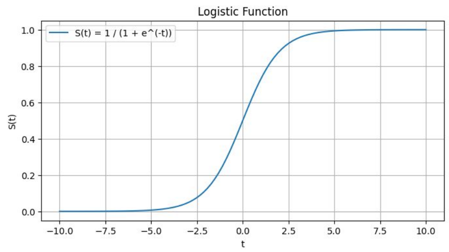
$$\text{logit}(P(Y = 1|X)) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

  - Thus, the logit function provides a direct link between the linear combination of features and the log-odds of a positive outcome.



- **p = 0.1**: Here, the probability is low, and the logit value is negative, indicating a decreased likelihood of the positive outcome.
- **p = 0.5**: This point corresponds to equal odds (logit = 0), meaning the likelihood of positive and negative outcomes is balanced.
- **p = 0.9**: At this high probability, the logit value is positive and large, signifying a higher likelihood of the positive outcome.

Logit vs Sigmoid



- **Sigmoid (Logistic) Function:**
  - The sigmoid function is the inverse of the logit function and converts the linear combination z into a probability between 0 and 1:

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

  - Substituting the linear combination into the sigmoid function, we can predict the probability of the positive outcome as:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

  - This relationship translates the linear combination of features and coefficients into a probability of the positive outcome by applying the nonlinear sigmoid transformation.

CHAPTER 3
CLASSIFICATION

CLASSIFICATION

LOGISTIC REGRESSION

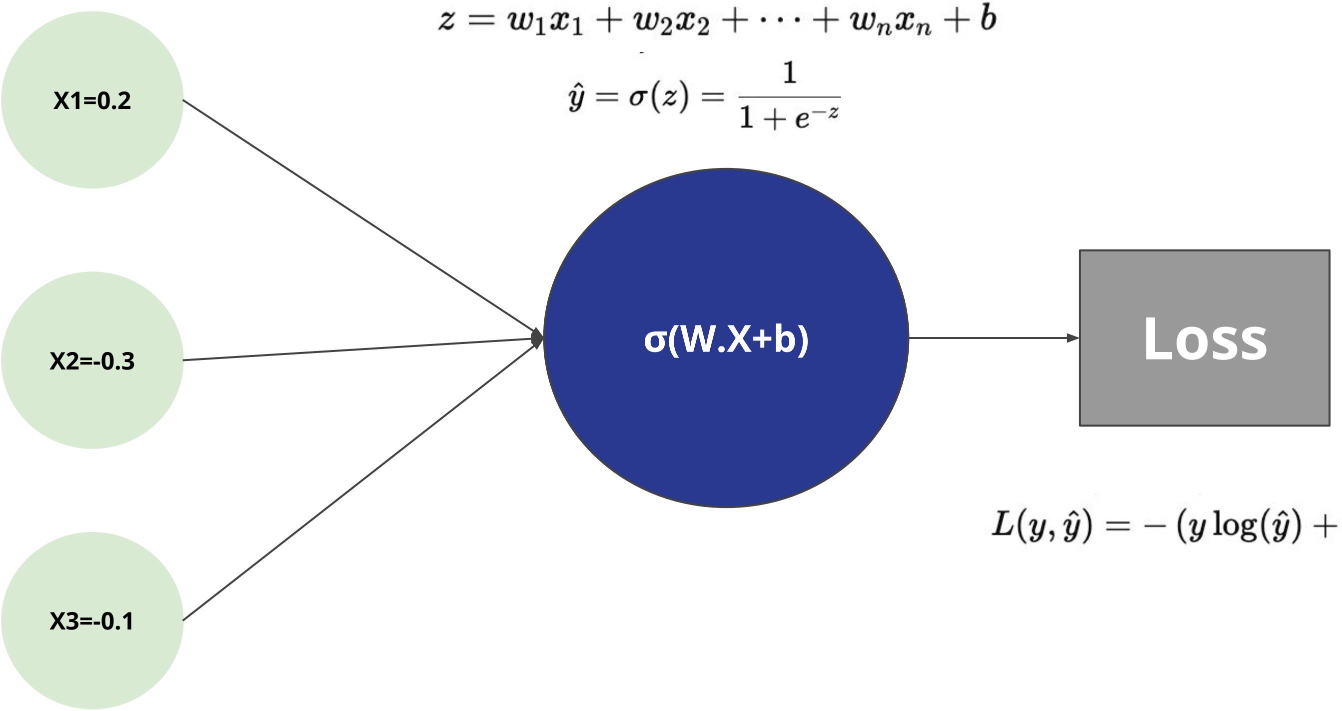
CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Logistic Regression as Perceptron

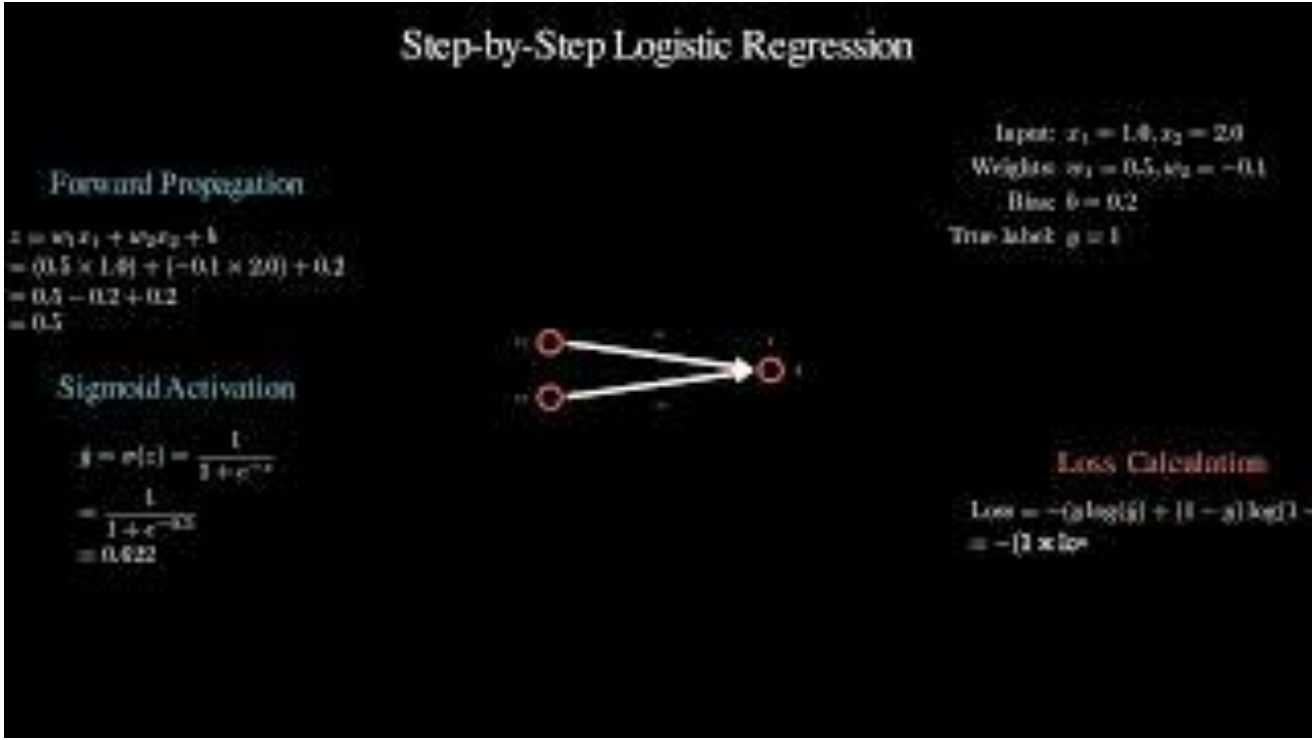
$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

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



$$L(y,\hat{y}) = -\left(y\log(\hat{y}) + (1-y)\log(1-\hat{y})\right)$$

Logistic Regression as Perceptron



Logistic Regression: Bridging Perceptron

Logistic Regression as a Perceptron

1. Input Features:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$

2. Weights and Bias:

$$\mathbf{w} = [w_1, w_2, \ldots, w_n], \quad b$$

3. Linear Combination:

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

- Weighted sum of input features plus bias.

4. Sigmoid Activation:

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

- Maps z to a value between 0 and 1 (probability).

Logistic Regression: Mathematical Formulation

Mathematical Representation of Logistic Regression

1. Linear Combination (Weighted Sum):

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

- Computes the linear combination of input features $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ and weights $\mathbf{w} = [w_1, w_2, \ldots, w_n]$, adding the bias b.

2. Sigmoid Activation Function:

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

- Transforms the linear combination z into a probability $\hat{y}$, mapping the result between 0 and 1.

3. Log-Loss Function:

$$L(y, \hat{y}) = -\left(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right)$$

- Measures the performance of a classification model whose output is a probability between 0 and 1. The loss increases as the predicted probability diverges from the actual label y.

Gradient Descent: Optimizing Weights and Bias

Optimizing Logistic Regression Model

Gradient Descent Method:

- **Purpose**: Minimize the loss function L to find the best parameters (w and b).
- **Update Rules:**

$$w_j := w_j - \eta \frac{\partial L}{\partial w_j}, \quad b := b - \eta \frac{\partial L}{\partial b}$$

  - η (eta) is the learning rate, controlling the size of the update step.
  - w_j are the weights, b is the bias.

Partial Derivatives:

- **Gradient with respect to weights:**

$$\frac{\partial L}{\partial w_j} = (y - \hat{y})x_j$$

- **Gradient with respect to bias:**

$$\frac{\partial L}{\partial b} = (y - \hat{y})$$

  - $\hat{y}$ is the predicted probability from the sigmoid function.
  - y is the actual label.
  - x_j is the input feature corresponding to weight w_j.

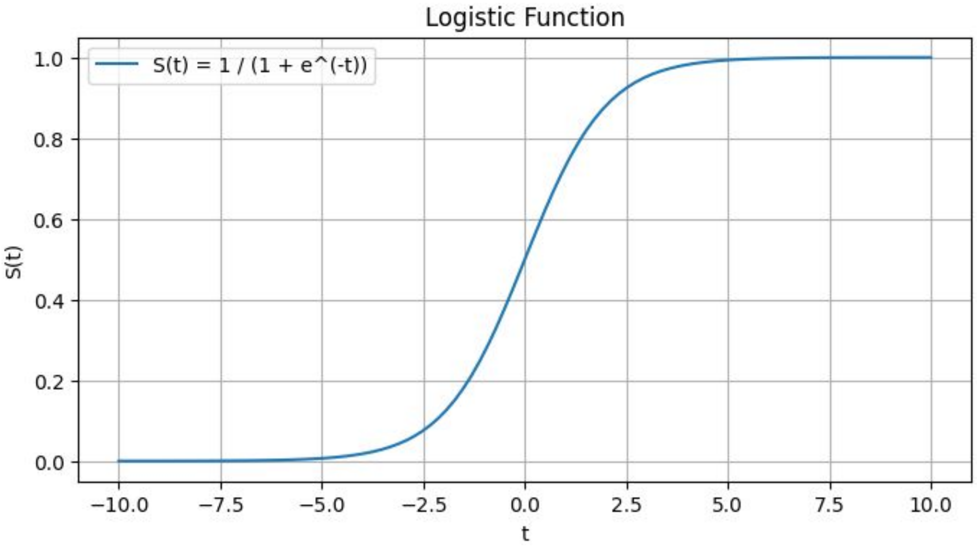
Sigmoid Function

- **Definition:**
  - Logistic Regression is a statistical method for binary classification that models the probability of a binary response based on one or more predictor variables.
- **Mathematical Representation:**
  - It estimates probabilities using the logistic function, which is an S-shaped curve that maps any real-valued number into the (0, 1) interval, making it suitable for modeling probability.
- **Logistic Function:**
  - The logistic function is defined as:

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

  - Where z is the linear combination of the input features X and their corresponding coefficients β, expressed as:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$



CHAPTER 3
CLASSIFICATION

CLASSIFICATION

TRAINING A LOGISTIC REGRESSION CLASSIFICATION MODEL

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Preparing the Heart Disease Dataset

Input Features **Output**

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num	Target
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0	NO
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2	YES
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1	YES
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0	NO
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0	NO

Separate Features from Target

- **Data Loading:**
  - Import dataset from a URL into a pandas DataFrame.
- **Target Variable Creation:**
  - Generate a new binary 'Target' column:
    - 'YES' for heart disease ($\mathrm{num} > 0$)
    - 'NO' for no heart disease ($\mathrm{num} = 0$)
- **Data Segmentation:**
  - Separate the dataset into features (X) and target (y).
- **Tools and Functions:**
  - `pd.read_csv()`: Reads data from URL.
  - `.apply()`: Applies a lambda function for binary classification.

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import log_loss
from sklearn.preprocessing import OneHotEncoder

# URL of the dataset
url = "https://www.riotu-lab.org/cs313/heart_data.csv"

# Read the dataset from the URL
df = pd.read_csv(url)

# Add a new column 'Response' based on the 'num' column
df['Target'] = df['num'].apply(lambda x: 'YES' if x > 0 else 'NO')

# Display the first few rows of the dataset
print("Sample dataset:")
#print(df.head())

# Separate features and target variable
X = df.drop(['num', 'Target'], axis=1)
y = df['Target']
```

One-Hot Encoding Categorical Features

- **Purpose:**
  - Convert categorical variables into a form that can be provided to machine learning algorithms to improve model performance.
- **Process:**
  - **Identify Categorical Features:**
    - List of categorical variables: 'sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'.
  - **Apply One-Hot Encoding:**
    - Use **`OneHotEncoder`** to transform these features into binary columns.
    - Handle unknown categories by ignoring them (**`handle_unknown='ignore'`**).
- **Integration:**
  - Create a new DataFrame **`X_encoded`** from the encoded data.
  - Name new columns using the original feature names to maintain clarity.
  - Merge encoded features back with the non-categorical features.
- **Key Functions Used:**
  - **`OneHotEncoder()`**: Transforms categorical values into a binary matrix.
  - **`fit_transform()`**: Fits the encoder to the data and transforms it.
  - **`get_feature_names_out()`**: Retrieves column names for the new binary matrix.
  - **`pd.concat()`**: Combines the original numeric features with the new encoded data.

```python
# Perform one-hot encoding on categorical features
categorical_features = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']
encoder = OneHotEncoder(handle_unknown='ignore')
X_encoded = pd.DataFrame(encoder.fit_transform(X[categorical_features]).toarray())
X_encoded.columns = encoder.get_feature_names_out(categorical_features)
X_encoded = pd.concat([X_encoded, X.drop(categorical_features, axis=1)], axis=1)
```

```python
[50]
X_encoded.columns = encoder.get_feature_names_out(categorical_features)
X_encoded.columns
```

```
Index(['sex_0', 'sex_1', 'cp_1', 'cp_2', 'cp_3', 'cp_4', 'fbs_0', 'fbs_1',
       'restecg_0', 'restecg_1', 'restecg_2', 'exang_0', 'exang_1', 'slope_1',
       'slope_2', 'slope_3', 'ca_0.0', 'ca_1.0', 'ca_2.0', 'ca_3.0', 'ca_nan',
       'thal_3.0', 'thal_6.0', 'thal_7.0', 'thal_nan'],
      dtype='object')
```

```python
X_encoded
```

	0	1	2	3	4	5	6	7	8	9	...	15	16	17	18	19	20	21	22	23	24
0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0
2	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	...	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
4	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
299	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
300	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
301	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
302	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0

Training and Testing the Logistic Regression Model

- **Data Splitting:**
  - **Objective:** Partition the data into training and testing subsets to evaluate the model's performance on unseen data.
  - **Method:**
    - Use `train_test_split` from scikit-learn.
    - Assign 20% of the data to the test set (`test_size=0.2`).
    - Ensure consistency across runs with `random_state=42`.
- **Model Training:**
  - **Logistic Regression Setup:**
    - Initialize the Logistic Regression model.
  - **Training Process:**
    - Fit the model on the training data using `model.fit(X_train, y_train)`.
- **Key Functions Used:**
  - `train_test_split()`: Splits the dataset into separate training and testing sets.
  - `LogisticRegression()`: Creates a logistic regression model.
  - `fit()`: Trains the logistic regression model on the training data.

```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
                                                    test_size=0.2,
                                                    random_state=42)

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

Evaluating Model Performance

- **Accuracy as a Performance Metric:**
  - **Purpose:** Evaluate the accuracy of the logistic regression model on the test dataset.
  - **Method:** `model.score(X_test, y_test)` calculates the accuracy, the proportion of correct predictions.
- **Code and Output:**
  - **Python Execution:**

python Copy code

```python
log_likelihood = model.score(X_test, y_test)  # Note: This is act
print("Final log-likelihood:", log_likelihood)
```

  - **Interpretation:** The reported score, labeled as 'log-likelihood', is actually the accuracy of the model, which is approximately 0.869. This means the model correctly predicts the outcome for 86.9% of the test cases.
- **Key Python Function:**
  - `model.score()`: Computes the accuracy, not the log-likelihood, contrary to what the variable name suggests.

```python
# Get the final log-likelihood
log_likelihood = model.score(X_test, y_test)
print("Final log-likelihood:", log_likelihood)
```

```
Final log-likelihood: 0.8688524590163934
```

CHAPTER 3
CLASSIFICATION

CLASSIFICATION

PREDICTION WITH A LOGISTIC REGRESSION MODEL

CS316: INTRODUCTION TO AI AND DATA SCIENCE

Prof. Anis Koubaa

Making Predictions with Logistic Regression

```python
# Make predictions on the test set
y_pred = model.predict(X_test)

# Display a sample of predictions
print("Sample predictions:")
pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).head(10)
```

Sample predictions:

	Actual	Predicted
179	NO	NO
228	YES	YES
111	YES	YES
246	YES	YES
60	YES	YES
9	YES	YES
119	YES	YES
223	YES	YES
268	YES	NO
33	NO	YES

- **Model Prediction:**
  - **Objective:** Use the trained logistic regression model to make predictions on the test dataset.
  - **Process:**
    - Predict using `` `model.predict(X_test)` ``, which classifies each instance in the test set based on the learned parameters.
- **Displaying Predictions:**
  - **Sample Output:**
    - Combine actual and predicted labels in a DataFrame to compare results visually.
    - **Code Snippet:**

python Copy code

```python
y_pred = model.predict(X_test)
sample_predictions = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).head
print("Sample predictions:", sample_predictions)
```

- **Key Functions Used:**
  - `` `model.predict()` ``: Generates label predictions for the given input data.
  - `` `pd.DataFrame()` ``: Constructs a DataFrame from the actual and predicted labels for easy comparison.

Making Predictions with Logistic Regression

- **Beyond Accuracy:**
  - What limitations does relying solely on accuracy present?
  - How might different thresholds affect model evaluation?
- **Understanding Errors:**
  - What can precision, recall, and F1-score tell us that accuracy cannot?
  - How does the confusion matrix provide deeper insights into model performance?
- **Handling Imbalance:**
  - How should we adjust our evaluation for imbalanced datasets?
  - What methods like SMOTE or cost-sensitive learning could be applied?
- **Testing Generalization:**
  - How robust is the model against new, unseen data?
  - What validation techniques ensure the model generalizes well?
- **Deep Dive into Model Assessment:**
  - Why is cross-validation critical for assessing model reliability?
  - How does the ROC-AUC curve help in understanding class differentiation?

END OF LECTURE 1