

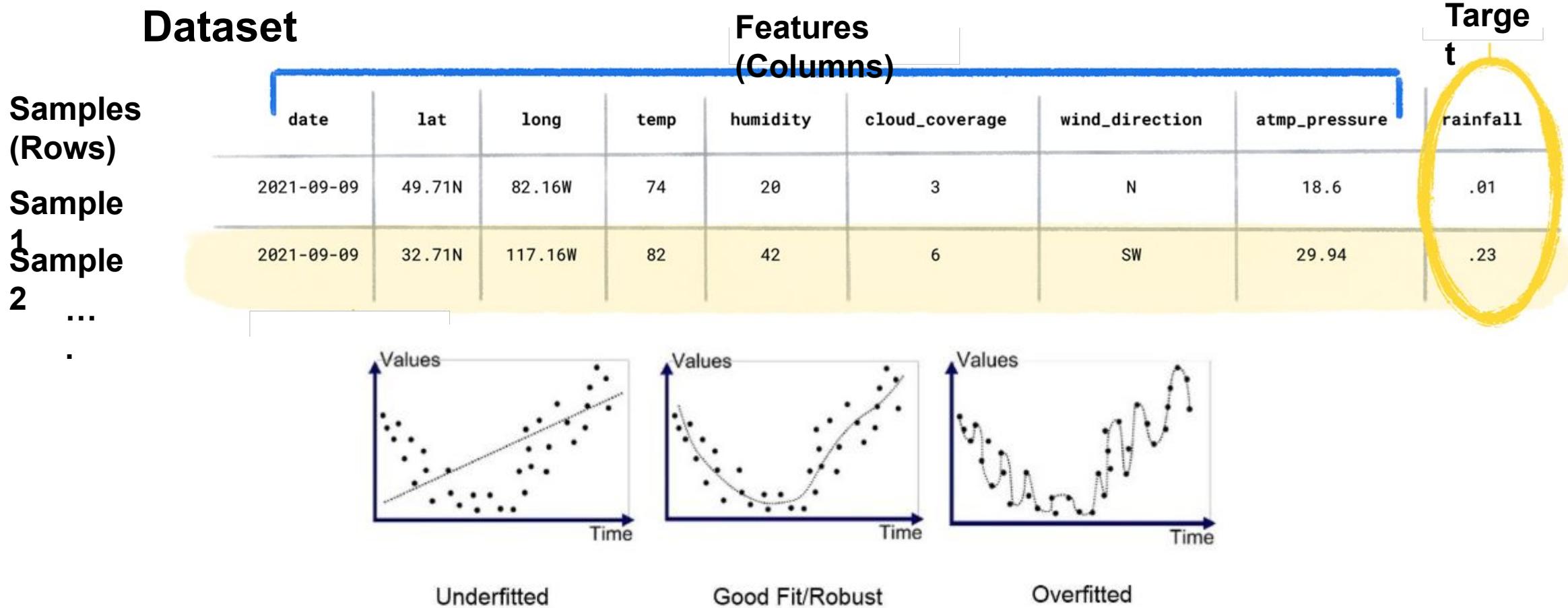
Machine Learning

Linear Regression Notes

Concepts in Machine Learning (ML)

- **Model:** An algorithm that describes/predicts an outcome, based on the observed data.
- **Fitting:** A model trained (fitted) in a dataset to learn patterns, relationships, or trends within that data.
- **Underfitting:** A model achieves poor performance in the training data.
- **Overfitting:** A model achieves great performance in the training (known) data, but it fails to generalize well in unseen (new) data.
- **Generalization:** The model's ability to make good predictions in both training and unseen data.
- **Weights/Coefficients:** The parameters of a model that need to be found by training it in a dataset.
- **Hyperparameters:** The settings of an ML algorithm.
- **Hyperparameter Tuning:** The process in which several settings (hyperparameters) are adjusted for a specific dataset.
- **Observations/Samples:** The number of paradigms (rows) in a dataset.
- **Features:** The attributes (columns) of an observation (e.g. the features of a weather dataset could include Temperature, Humidity, Atmospheric Pressure, etc.).
- **Labels/Targets:** The feature, whose value is unknown and needs to be predicted.

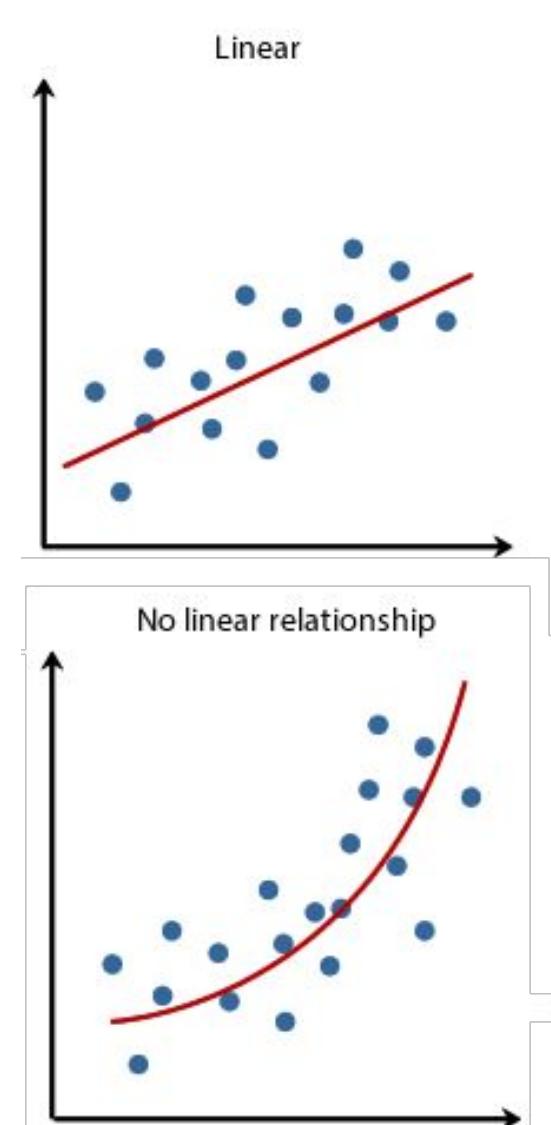
ML Concepts Example



A model that is trained to estimate the ***rainfall*** over the time.

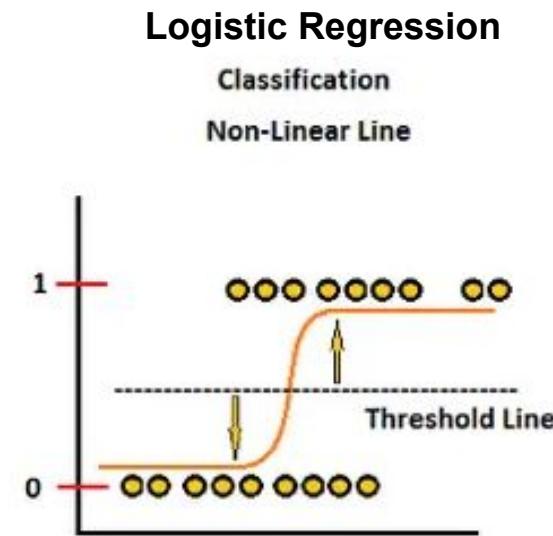
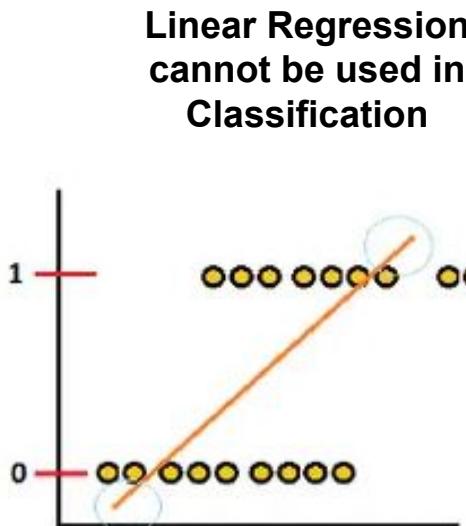
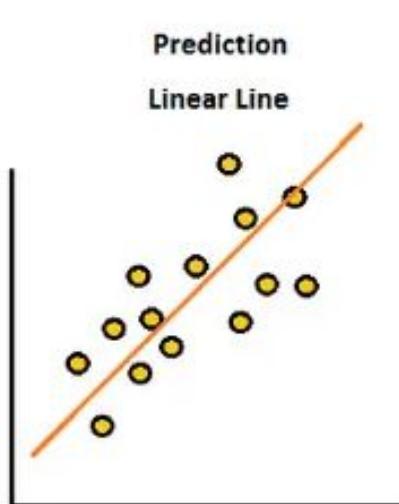
Regression Analysis

- Estimates the relationship between one or more independent variables (**features**) X and one dependent variable (**target**) y , by calculating some parameters w .
- Alternatively, it estimates a function: $f(w, x) = y$.
 - $f(w, x)$ might not always be accurate. It tries to approximate y as best as possible.
- Two main types of Regression Analysis:
 - **Linear:** The model assumes linear relationship between inputs x and outputs y .
 - **Non-Linear:** The model assumes non-linear relationship between x and y .
- The regression types can be furthered categorized in:
 - **Univariate:** The target value $y \in R$ is a single value.
 - **Multivariate:** The target value $y \in R^D$ consists of D dependent variables.



Linear Regression

- Statistical Process.
- Linear type of Regression.
- Very powerful algorithm, used in medicine, biology, finance, etc. **Why?**
- Two main types of Regression Analysis:
 - **Linear** (The target is $y \in R$). Used in **Regression** tasks.
 - **Logistic** (The target is a binary label $y \in \{0, 1\}$). Used in **Binary Classification** tasks.



Linear Modeling

- **Question:** Given a vector of inputs $x = (x_1, x_2, \dots, x_i)$, how can I combine them to compute the target y ?
 - Example: Given the $x = (x_1: \text{Floors}, x_2: \text{Number of Rooms}, x_3: \text{Size})$ of a house, how can I calculate its price y ?
- **Linear Regression Hypothesis:** There is a linear relationship between x and y . Each variable x_i can contribute to calculate y as follows:

$$w_1 \cdot \text{Floors} + w_2 \cdot \text{Number of Rooms} + w_3 \cdot \text{Size} + w_0 = \text{Price}$$

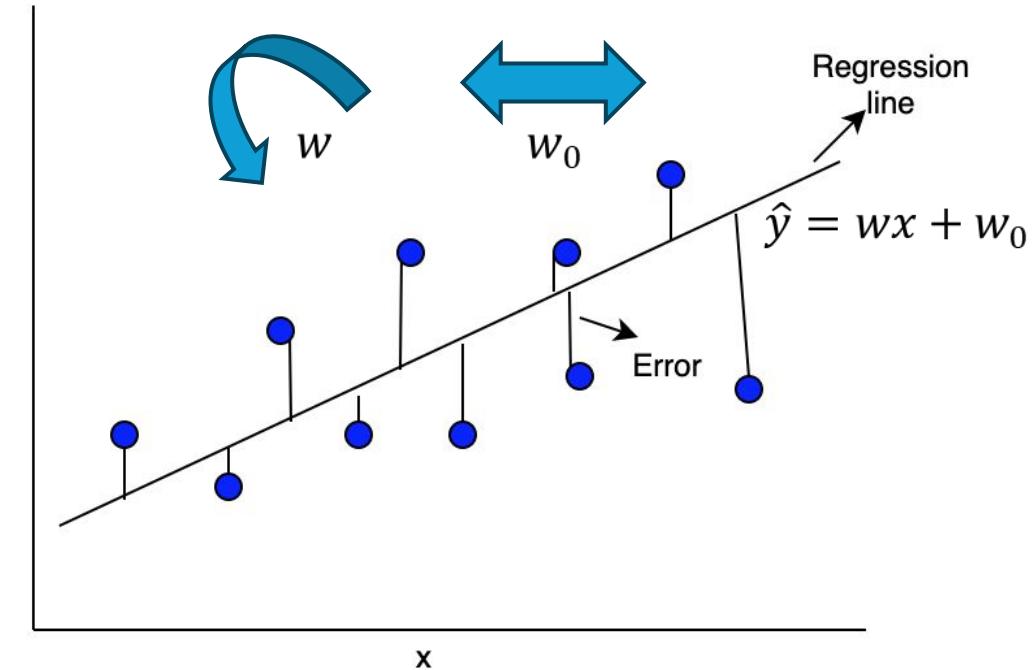
- Each **coefficient (weight)** w_i shows how much each variable x_i contributes to the outcome. For instance:

- For $w_1 = 5000$, it means that each floor increases the price of the house by 5000€.
- For $w_2 = 10000$, it means that each room costs 10000€.
- For $w_3 = 1000$, it means that each squared meter of the house costs 1000€.
- For $w_0 = 10000$ means that the price starts from 10000€ independently.
- For house with 2 floors, 3 rooms and $100 m^2$, the price of the house can be calculated as:

$$y = 5000 \cdot 2 + 10000 \cdot 3 + 1000 \cdot 100 + 10000 = 150000\text{€}$$

Linear Regression with 1 Feature

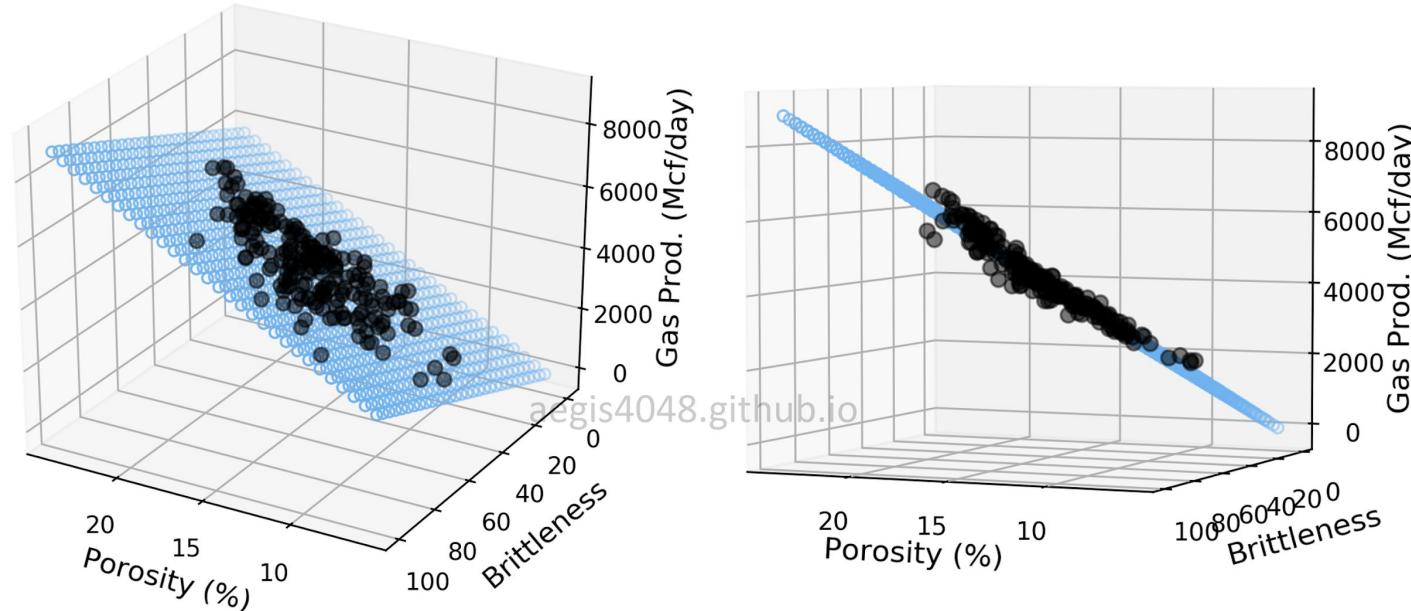
- Let there be a dataset (X, y) , where $y \in R$.
- The goal is to find a linear function $f(x) = y$ for every given **training point (sample)** x .
- f can be calculated as a **Regression Line** L : $\hat{y} = ax + b$ or alternatively: $\hat{y} = wx + w_0$, where \hat{y} is the estimated (predicted) y value.
- The **weight (coefficient)** w controls the **orientation (rotation)** of the line.
- The weight w_0 (**bias**) affects the **position (left-right)** of the line.
- The goal is to find $W: \{w, w_0\}$ so that for every x , the line fits perfectly.
- Notice that the line does not fit perfectly the data. Each prediction \hat{y}_i generates an error e_i , which can be calculated as $e_i = (y_i - \hat{y}_i)^2$



Linear Regression with Many Features

- The same concept can be applied to multiple features, where X has multiple features.
- In these cases, the linear regression constructs a **hyperplane** (instead of a line) that fits the data.
- For dataset with D features, formula becomes: $\hat{y} = w_1x_1 + w_2x_2 + \dots + w_Dx_D + w_0 = W^T X + w_0$, where the goal is to find w_i that result in a perfect (as good as possible) hyperplane.

3D multiple linear regression model



Linear Regression Solution

- Let $\hat{y} = wx$ be the predicted value of the actual target y .
- The error of the
- Goal is to adjust W so that the **Squared Error** $(\hat{y}_i - y_i)^2$ is minimized for each sample x_i .
- To fit the regression line for every sample x_i , the **Mean Squared Error (MSE)** must be minimized.
- $MSE(\hat{y}, y) = 0 \Rightarrow \sum_{i=1}^N (\hat{y}_i - y_i)^2 = 0 \Rightarrow \sum_{i=1}^N (W^T X - y_i)^2 = 0$.
- **Popular Optimization Methods (Solvers):**
 - Ordinary Least Squares (OLS).
 - Gradient Descent (GD).
 - Stochastic Gradient Descent (SGD).
 - Stochastic Average Gradient (SAGA).
 - Limited-memory BFGS (LBFGS).

Optimization Methods (1)

- **Ordinary Least Squares (OLS)**

- Analytical solution.
- Provides the optimal solution for W .
- Time complexity is $O(ND^2 + D^3)$, where D is the number of features and N is the number of training samples.
- Computational expensive for large datasets (million samples with many features).

- **Gradient Descent (GD)**

- Approximated solution.
- First-Order Iterative algorithm.
- Time complexity is $O(DNK)$, where K is the number of training iterations (**epochs**).
- Can approximate the solution in large datasets (million samples), without expensive hardware.

- **Stochastic Gradient Descent (SGD)**

- Like GD, but it updates the weights using batches (subsets of the training set).
- Faster than GD, because it can be slow to calculate the gradients for the entire training set.
- Time complexity is $O(\frac{DNK}{S})$, where S is the batch size.

Optimization Methods (2)

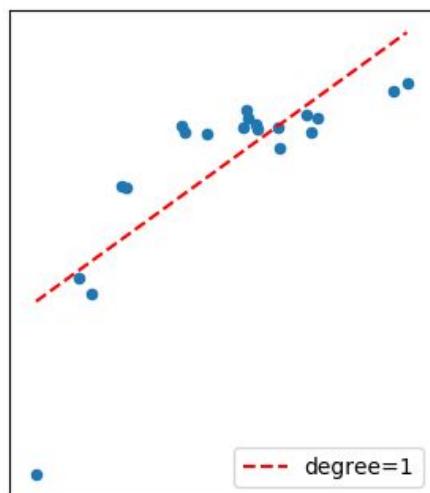
- Stochastic Average Gradient (SAGA)
 - Alternative to SGD, which usually converges faster for simple linear models, such as Linear/Logistic Regression.
- Limited-memory BFGS (LBFGS)
 - Approximated solution.
 - Second-Order Iterative algorithm.
 - Time complexity is $O(DNK)$, where K is the number of training iterations (**epochs**).
 - Converges faster than GD or SGD, but usually to sub-optimal solutions.

Polynomial Regression

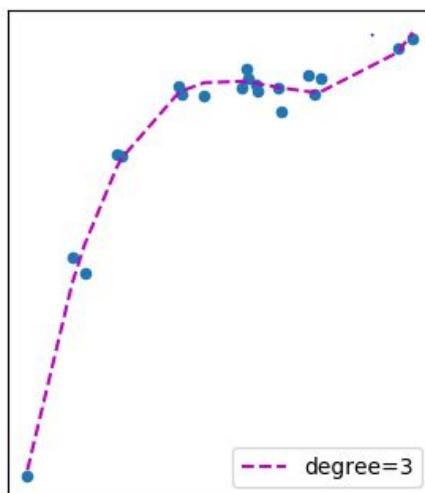
- Let D be the number of features. The polynomial Regression is defined as:

$$y = w_0 + w_{11}x_1 + \cdots + w_{1D}x_D + w_{21}x_1^2 + \cdots + w_{2D}x_D^2 + \cdots + w_{k1}x_1^k + \cdots w_Dx_D^k$$

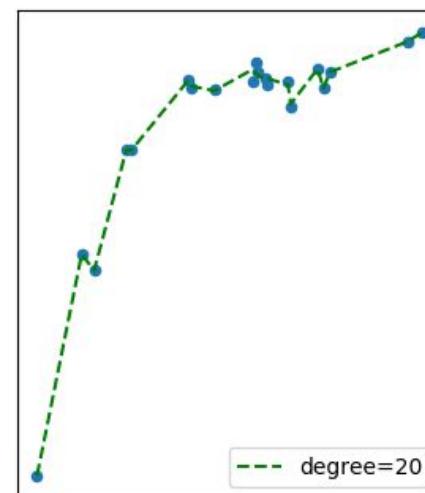
- Polynomial Regression can capture non-linear relationships between input X and target y .
- The order k will have to be adjusted (fine-tuned). Low k values could lead to underfitting issues, while large ones could lead to overfitting. Typical values: $k = \{3, 5\}$.**



Underfit
High Bias
Low Variance



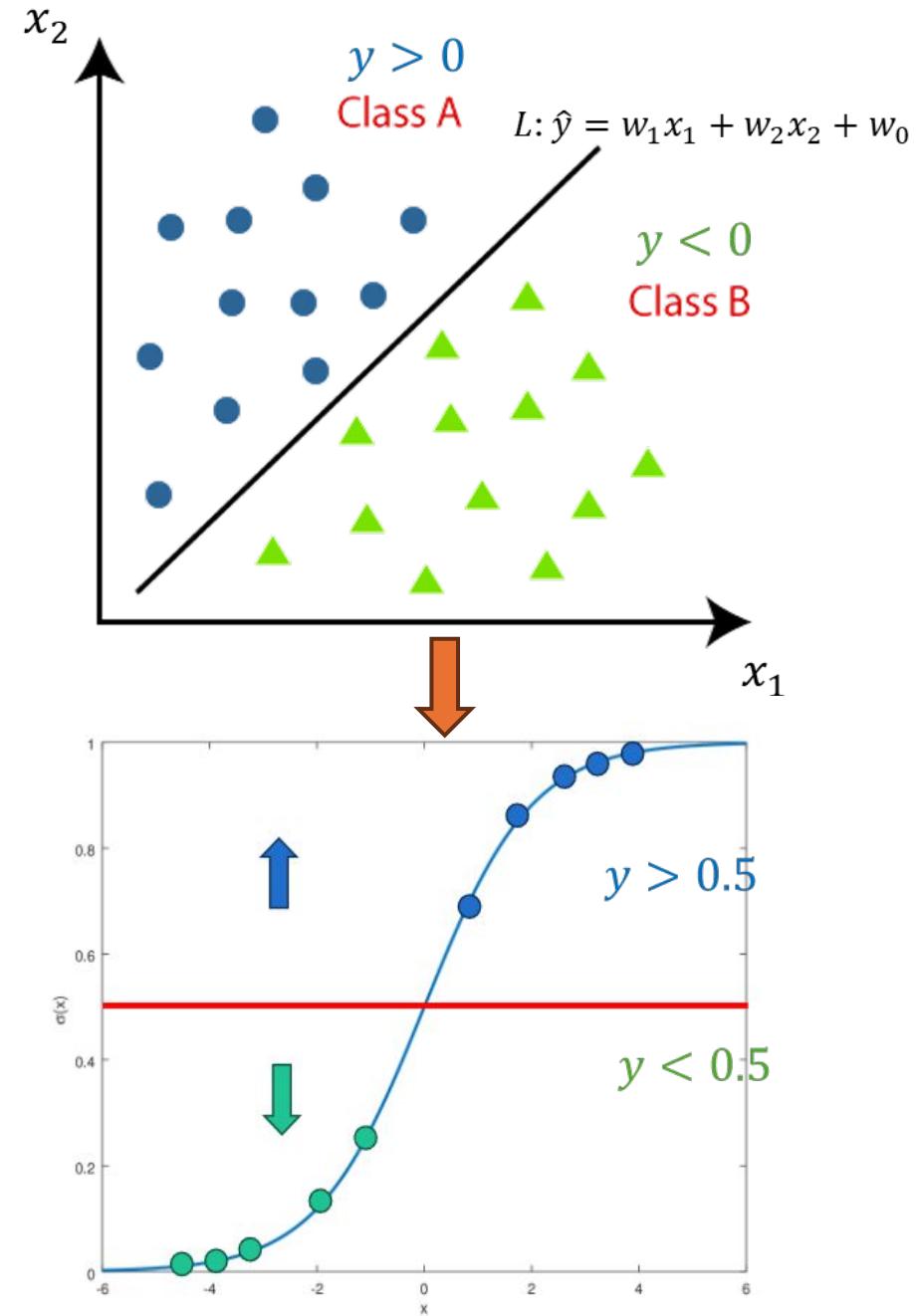
Correct Fit
Low Bias
Low Variance



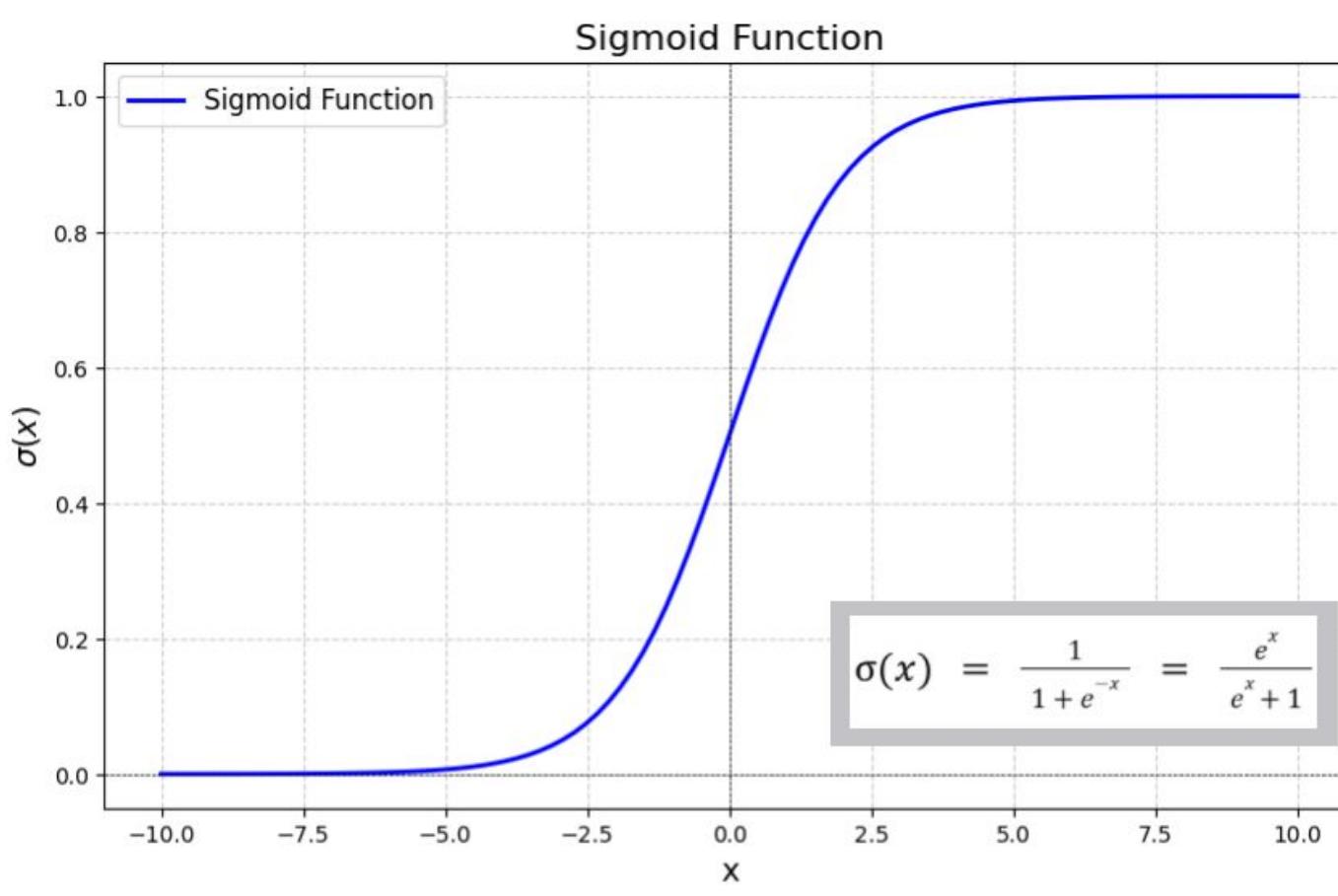
Overfit
Low Bias
High Variance

Logistic Regression

- Let there be a dataset (X, y) , where $y \in \{0,1\}$ with **labels** $c = \{0, 1\}$ representing classes A and B, respectively.
- The goal is to find a function $f(w, x)$ that separates A, B.
 - Alternatively, $f(w, x)$ should equal 0 if class belongs to the first class else 1.
- $f(w, x)$ can be calculated as a Regression line $L: \hat{y} = w_1x_1 + w_2x_2 + w_0$
- Given a sample $x = (x_1, x_2)$ and weights $W = (w_1, w_2, w_0)$, then:
 - $y = 0$ if the sample is placed on the line.
 - $y < 0$ if the sample is placed below the line.
 - $y > 0$ if the sample is placed above the line.
- Sigmoid (Logistic) Function:**
 - Converts \hat{y} to probability: Negative values become probability close to 0, while positive values become probability close to 1.0.
 - Formula: $\hat{c} = \sigma(\hat{y}) = \frac{1}{1+e^{-\hat{y}}}$



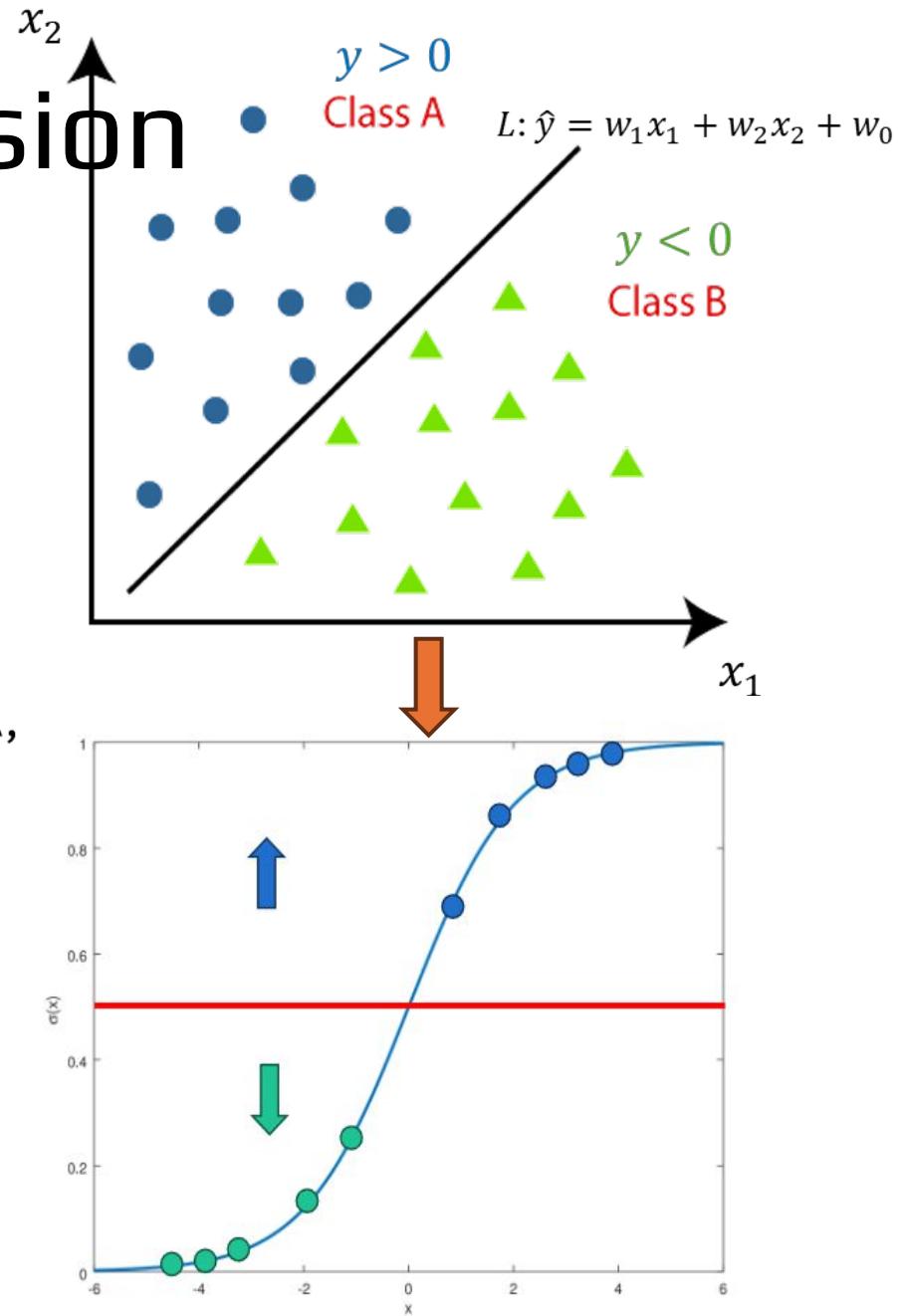
Sigmoid (Logistic) Function:



- For $x < 0$, $\sigma(x)$ becomes positive and close to 0.0 ($c = 0$).
- For $x = 0$, $\sigma(x) = 0.5$
- For $x > 0$, $\sigma(x)$ becomes positive and close to 1.0 ($c = 1$).

Optimizing Logistic Regression

- Logistic Regression is similar as Linear Regression but passes \hat{y} through the sigmoid function $\hat{c} = \sigma(x)$ to predict the label c .
- $MSE(\hat{y}, y) = 0 \Rightarrow \sum_{i=1}^N (\hat{y}_i - y_i)^2 = 0 \Rightarrow$
- $\sum_{i=1}^N (\sigma(WX) - y_i)^2 = 0$
- In this example: $\sum_{i=1}^N \left(\frac{1}{1+e^{-(w_1x_1+w_2x_2+w_0)}} - y_i \right)^2 = 0$
- **Optimizers (solvers)** that minimize MSE : (L-BFGS, SAG/SAGA, GD/SGD).
- An improved version of Logistic Regression minimizes **Binary Cross-Entropy(log-loss) BCE** (y, \hat{y}), instead of MSE .
- $BCE(y_i, \hat{y}_i) = -\frac{1}{N} \sum_{i=1}^N [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$
 - Av $y_i = 0$ (Κλάση $c = 0$), τότε $BCE(y_i, \hat{y}_i) = -(1 - y_i) \log(1 - \hat{y}_i)$.
 - Av $y_i = 1$ (Κλάση $c = 1$), τότε $BCE(y_i, \hat{y}_i) = y_i \cdot \log(\hat{y}_i)$.



Regularization

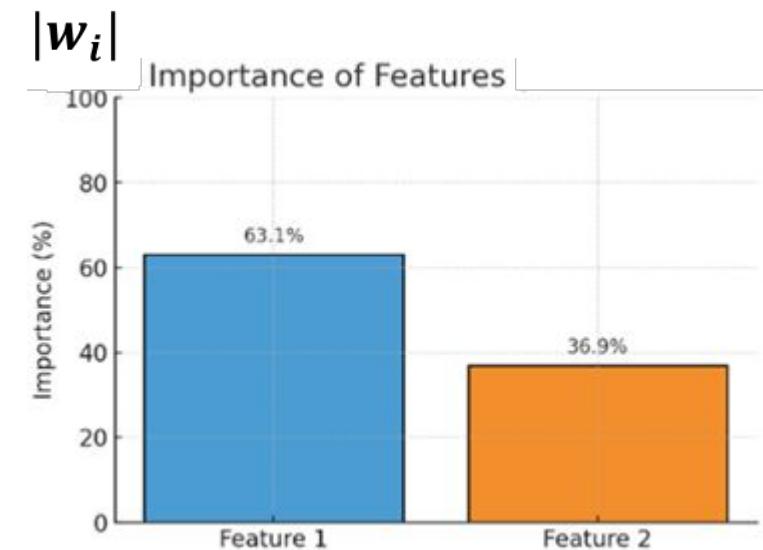
- In linear/logistic regression, a solver minimizes $loss(\hat{y}, y)$ (e.g. $MSE(\hat{y}, y)$ or $BCE(\hat{y}, y)$) by adjusting the coefficients W .
 - By default, the solver aims to adjust W so that $loss(\hat{y}, y) = 0$.
- However, if the computed weights are large, small changes in values of X could dramatically change the result. As a result, although it perfectly fits the training data, it could make mistakes in unknown samples (**overfitting**).
- **Question:** Is there a problem if $w_0 = 100, w_1 = 10000, w_2 = 0.01, w_3 = 0.02$?
 - Small changes in x_1 result in a completely different \hat{y} .
- Ideally, W values should be as small and uniform (in the same range).
- **Solution: Minimize both $loss(\hat{y}, y)$ and W at the same time.**

Regularization Penalties

- Let $W = (w_1, w_2, \dots, w_D, w_0)$.
- **L_2 Regularization**
 - Minimizes $loss(\hat{y}, y) + \lambda \sum_{i=0}^D w_i^2$, $0 \leq \lambda \leq 1$
 - Encourages all weights w_i to be calculated equally and avoid overfitting.
- **L_1 Regularization**
 - Minimizes $loss(\hat{y}, y) + \lambda \sum_{i=0}^D |w_i|$, $0 \leq \lambda \leq 1$
 - Encourages the least important weights w_i to become zero, highlighting the important features (**Feature Importance**).
 - By assigning a weight $w_i \approx 0$, the feature x_i is eliminated from WX equation (as $w \cdot x \approx 0$ if $w \approx 0$).
- **Elastic Net**
 - Minimizes $loss(\hat{y}, y) + \alpha(\lambda \sum_{i=0}^D |w_i| + \frac{1-\lambda}{2} \sum_{i=0}^D w_i^2)$, $0 \leq \lambda \leq 1$, $0 \leq \alpha \leq 1$
 - Attempts to address overfitting while highlighting the important features of the training data.

Coefficient Analysis

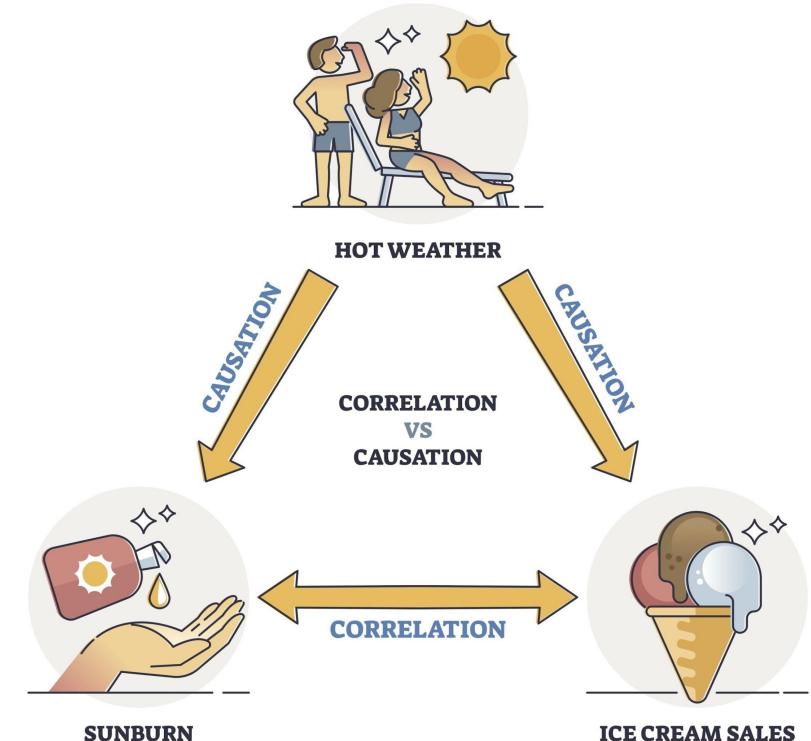
1. Normalize each feature x_i into range $[0.0, 1.0]$ using Min-Max formula: $x'_i = \frac{x_i - x_{i\min}}{x_{i\max} - x_{i\min}}$, where $x_{i\min}$ is the minimum value of x_i feature and $x_{i\max}$ is the maximum value.
2. Train a linear regression model with $L1$ regularization penalty.
3. Compute the absolute values of the coefficients.
4. Plot the coefficients using a bar plot.



Παράδειγμα για Linear
Regression με $D = 2$ features

Correlation vs Causation

- **Correlation:** Two variables appear to change synchronized.
 - For example, one might decrease as the other increases or vice versa.
- **Causation:** One variable ***directly influences*** another (causes it to increase or decrease).
 - For instance, one variable increases ***because*** the other decreases.
- **Causation ≠ Correlation**
- For instance, during summer, it is possible to eat an ice cream in the beach and get a sunburn. However, just because we eat ice cream doesn't mean we get sunburn! It's because of the sun!



Machine Learning

Linear Regression Notes