Introduction to Machine Learning with Python

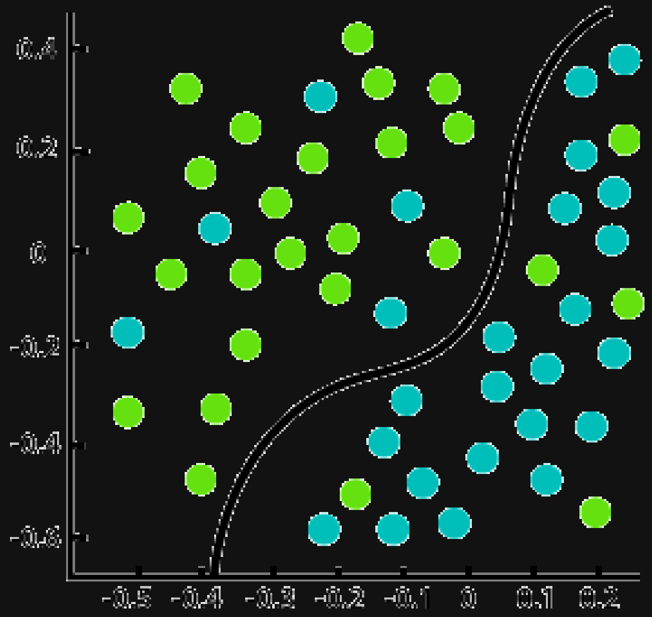
David Schaupp | WS2025

Content

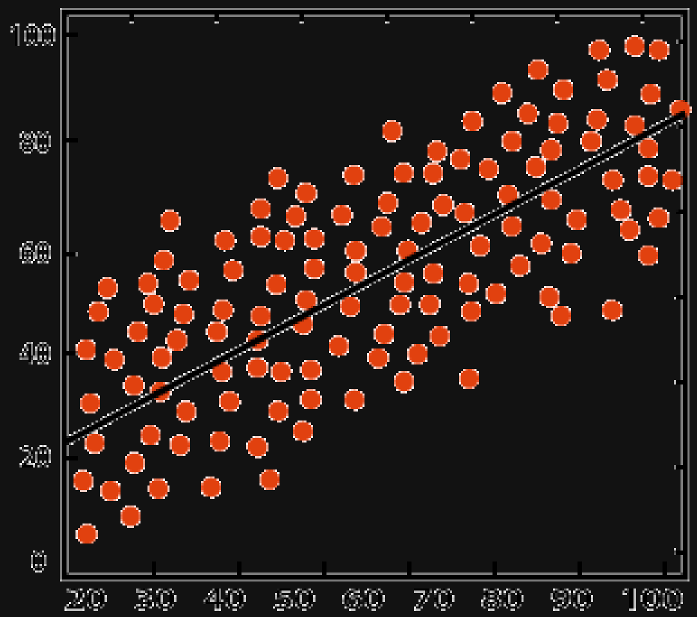
Regression vs Classification

```
Comparison:
- Type of Prediction
  Regression: Predicts continuous values
  Classification: Predicts discrete values
- Example
  Regression: Predicting the price of a house in Vienna
  Classification: Predicting if a house is in Vienna or not
- Task of Algorithm
  Regression: Mapping input value (x) with continuous output variable (y)
  Classification: Mapping the input value of x with the discrete output variable of y
- Problem Solution
  Regression: Solves regression problems such as house price predictions and weather predictions
  Classification: Solves classification problems like identifying spam e-mails, spotting cancer cells
                  and speech recognition
- Further Division
  Regression: Can be divided into Linear and Non-linear Regression
  Classification: Can be divided into Binary Classifiers and Multi-class Classifiers
```

Regression vs Classification



Classification



Regression

Linear Regression

```
Problem:
- Predict the price of a house, given the size of the house
Real Estate Agent Question:
- How much should I sell a 1250 sqft house for?
Solution:
- Fit a "straight line" (linear model) through the training data
Result:
- Intersect the line with 1250 sqft on x-axis
- Get estimated price on y-axis
```

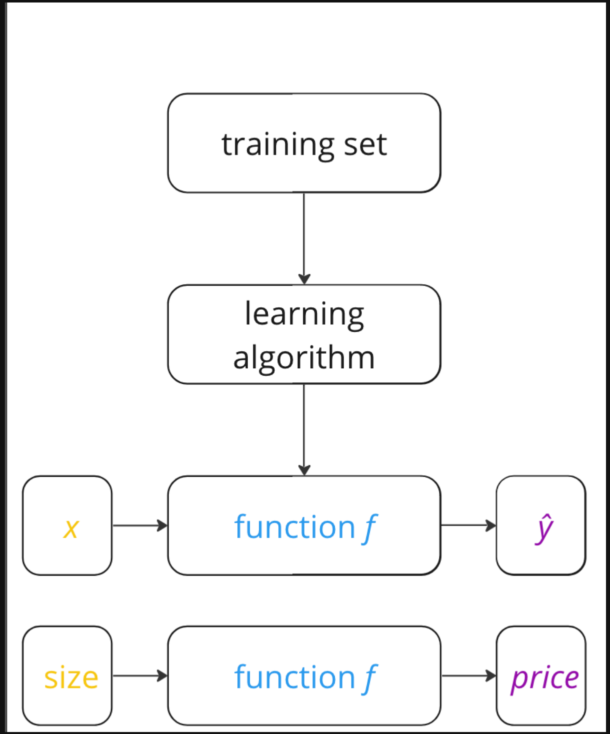
Linear Regression - Terminology

```
Notation:
  - Dataset: Data used to train the model
  - x: "input" variable/features (size in feet^2)
  - y: "output" variable/target variable
        (price in $1000´s)
  - m: Number of training examples (size of the dataset)
  - (x,y): One training example
    - (x^(i),y^(i)): i-th training example
    - First training example: (x^(1),y^(1)) = (2104,400)
```

size in feet^2	price in $1000's
2104	400
1416	232
1534	315
852	178
...	...
3210	870

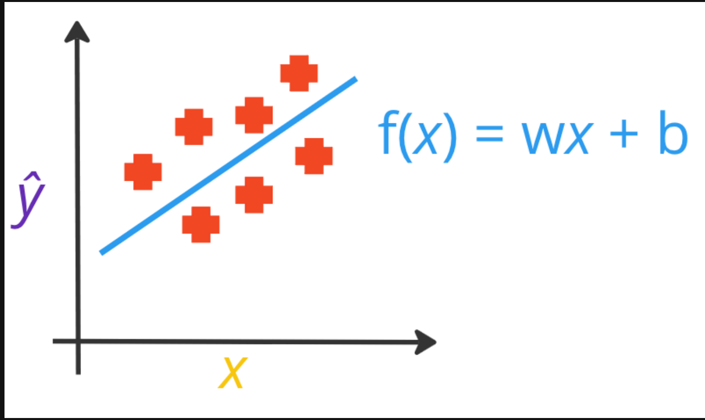
Linear Regression - Training Workflow

```
Training set:
  - features x
  - targets y
Learning algorithm:
  - Plug in x and y to learn patterns
Algorithm produces function f:
  - Takes new input x
  - Outputs prediction/estimate "y-hat"
House price example:
  - Plug size (x) into function/model
  - Get price (y-hat) from the function/model
```

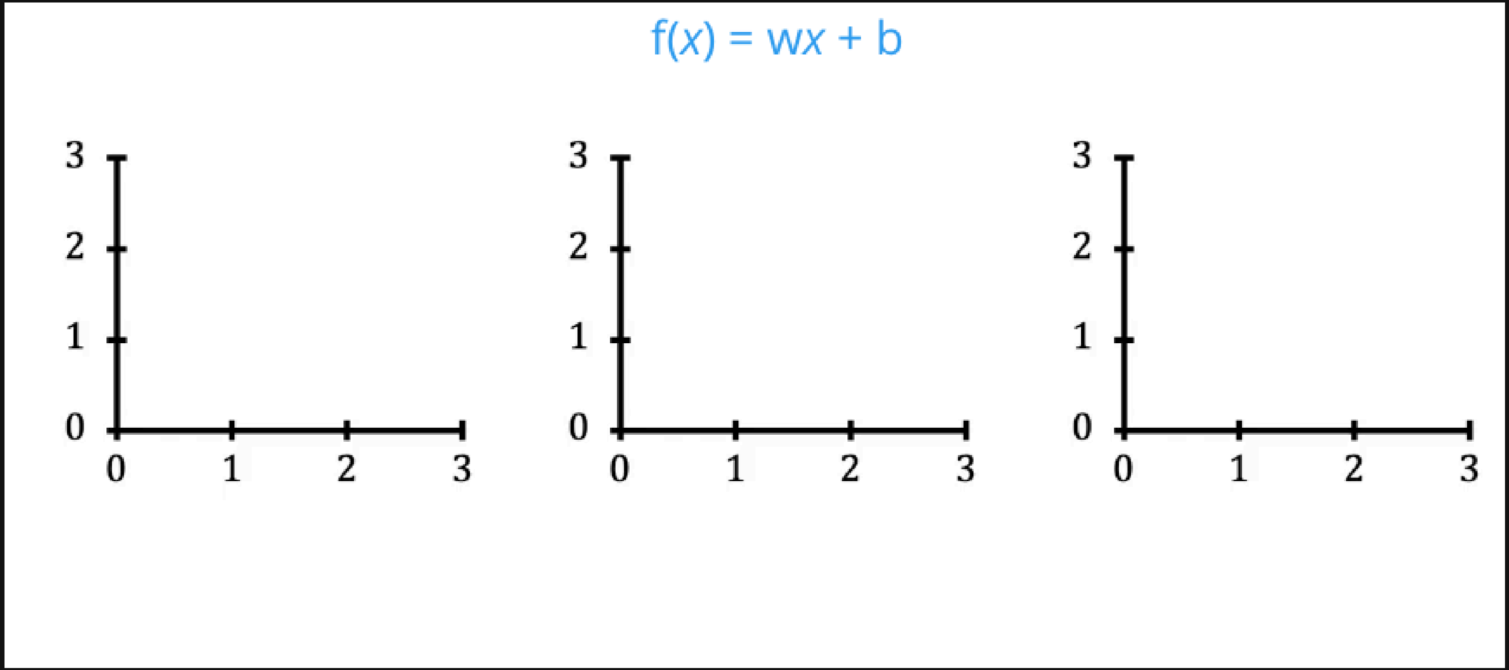


Linear Regression - How to represent f?

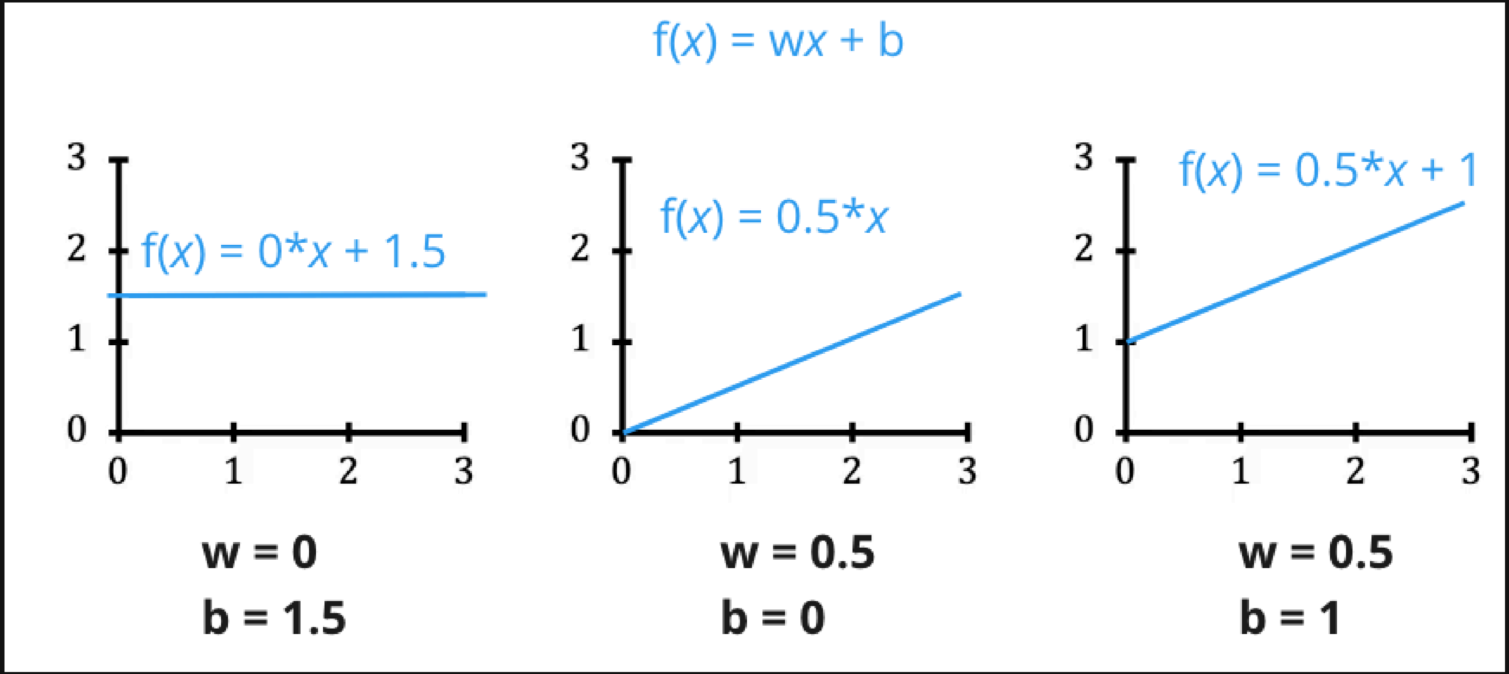
```
Model f:
  - Linear function
  - f(x) = wx + b
  - w: weight
  - b: bias
    - w and b are parameters of the model
```



Linear Regression - How to represent f: Example



Linear Regression - How to represent f: Example



Linear Regression - How to train?

```
Model training:
  - Setting the parameters w and b so that the model best fits the training data

Measure the fit of the model:
  - Cost function

Cost function for linear regression:
  - Squared Error Cost Function (Loss Function)
```

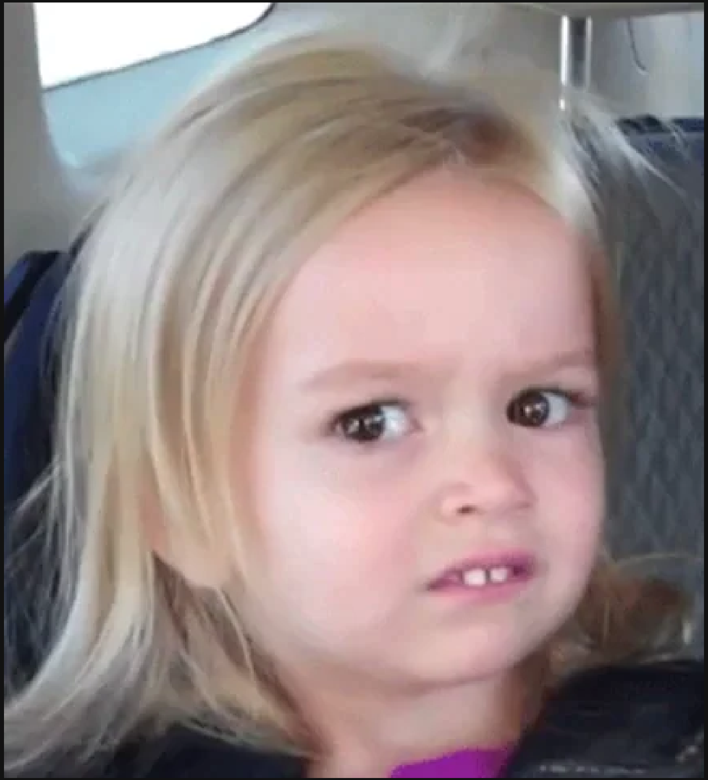
$$J_{(w,b)} = \frac{1}{m} \sum_{m}^{i=1} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

Linear Regression - Squarred Error (Construction)

$$J_{(w,b)} = \frac{1}{m} \sum_{m}^{i=1} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

```
Formula Construction Steps:
1. Measure error: difference between prediction y-hat^(i) and actual y^(i)
2. Square the error: make it positive
3. Sum across all training examples: total error
4. Divide by m: average error
5. Divide by 2: mathematical convenience for gradient calculation
```

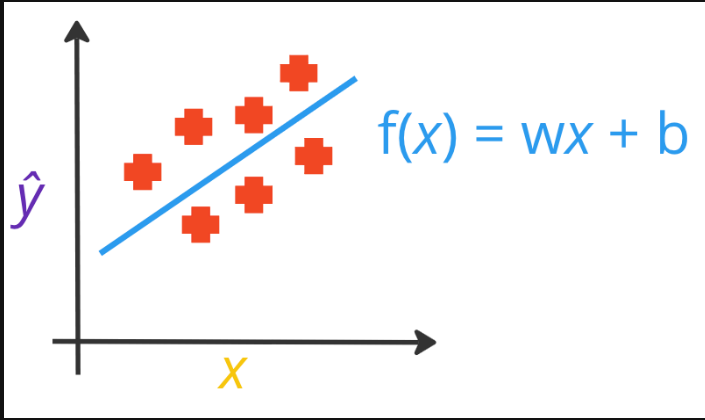
Enough maths (for now?)



Linear Regression - Training

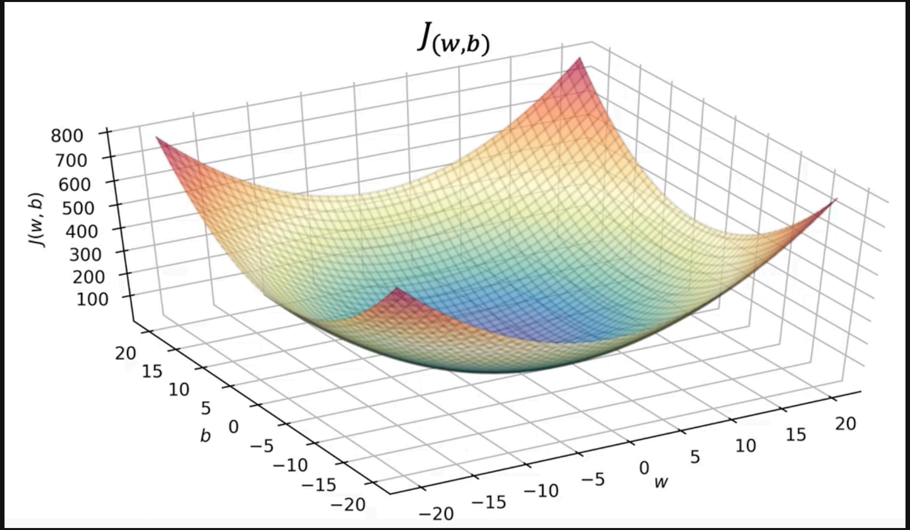
```
How to estimate the best parameters w and b for an
given dataset?
- Minimize the cost function

How to minimize the cost function?
- Gradient descent
```



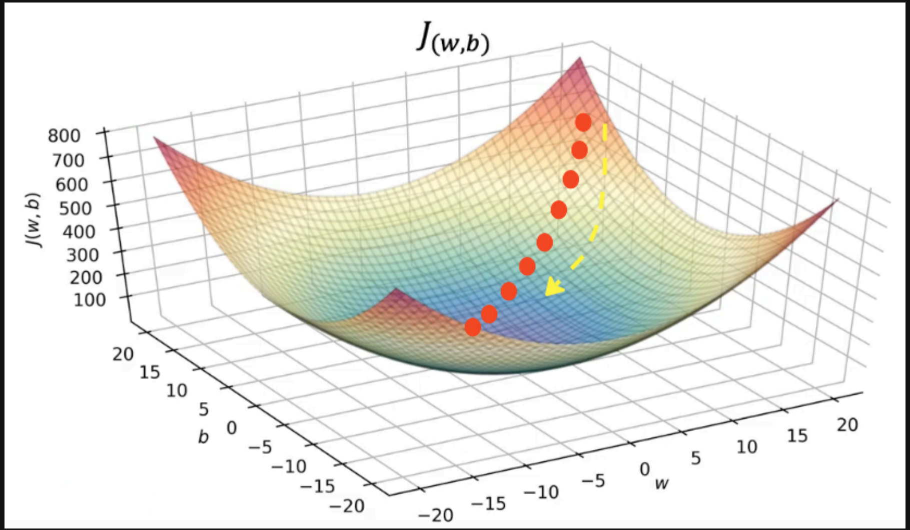
Linear Regression - Training

```
Squarred Error Cost function:
  - 3D surface plot: axes labeled w and b, height is cost J
  - Convex function (bowl shaped) - only one global minimum
  - Gradient descent will find the global minimum
  - Goal: Find parameters w and b that minimize J(w,b)
```



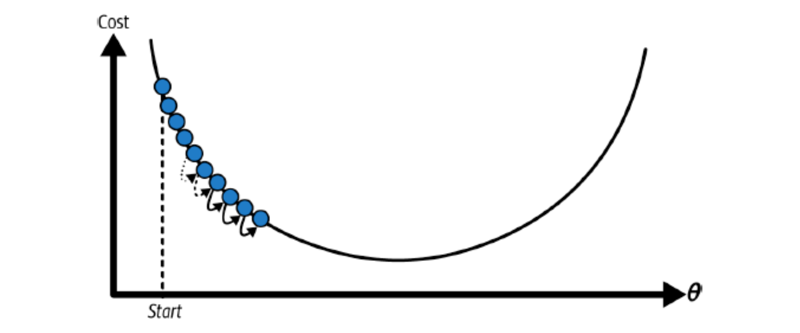
Linear Regression - Training

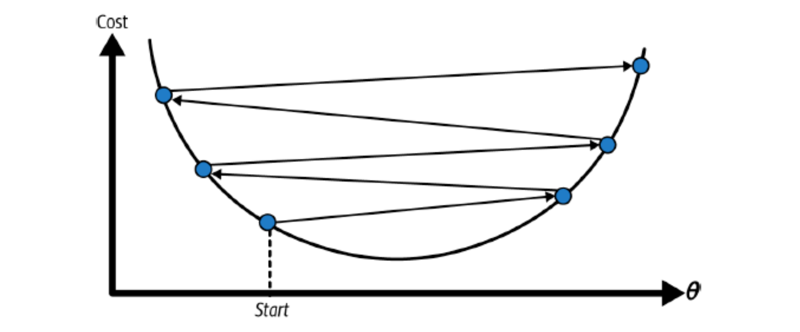
```
Gradient Descent:
  - Optimization algorithm to find optimal solution
  - General idea: tweak parameters iteratively to minimize the cost function
  - Steps:
    - Start with random parameters of w and b (random initialization)
    - Predict y-hat for each training example and calculate the error
    - Calculate the gradient of the cost function
    - Update the parameters in the opposite direction of the gradient
    - Repeat until convergence (until the gradient is zero)
```



Linear Regression - Training

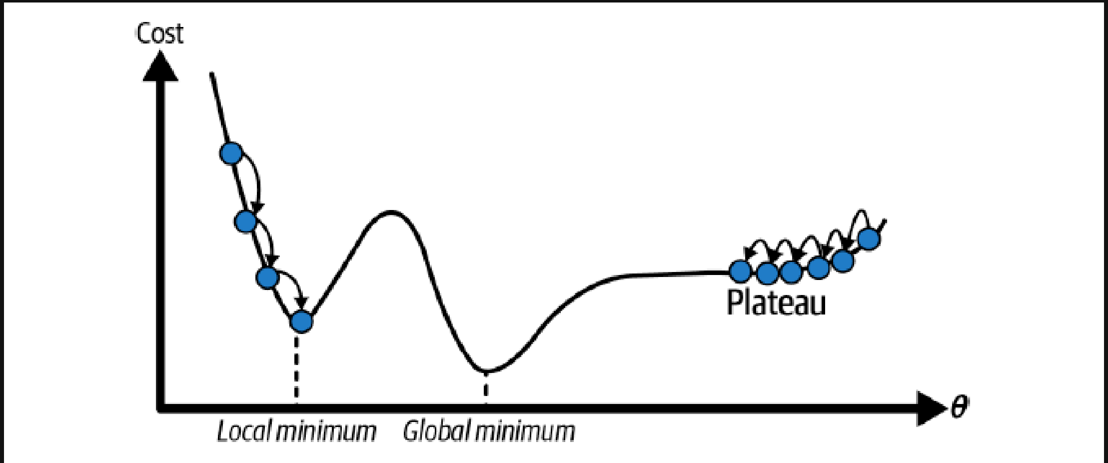
```
Learning Rate (Step Size):
  - Critical hyperparameter
  - Too small:
    - Slow convergence, long training
  - Too large:
    - Risk of overshooting minimum
```





Linear Regression - Training

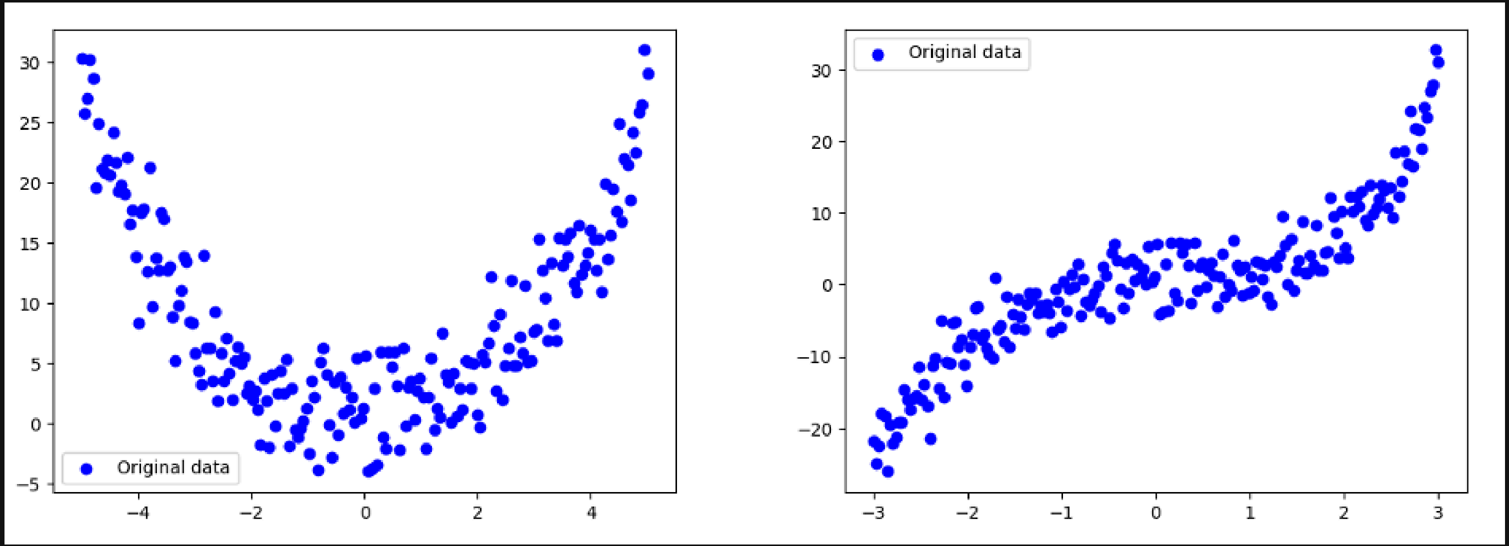
```
Gradient Descent:
  - Not all functions look like nice bowls
  - There are ridges, plateaus, etc.
  - Convergence is not that easy
  - If the random initialization starts the algorithm on the left:
    - The algorithm will converge to the local minimum
  - If the random initialization starts the algorithm on the right:
    - It will take long time to cross the pleateau
    - If training stops too early, the global minimum is not found
```



Linear Regression - Training

Polynomial Regression

```
How to Deal with such data:
```

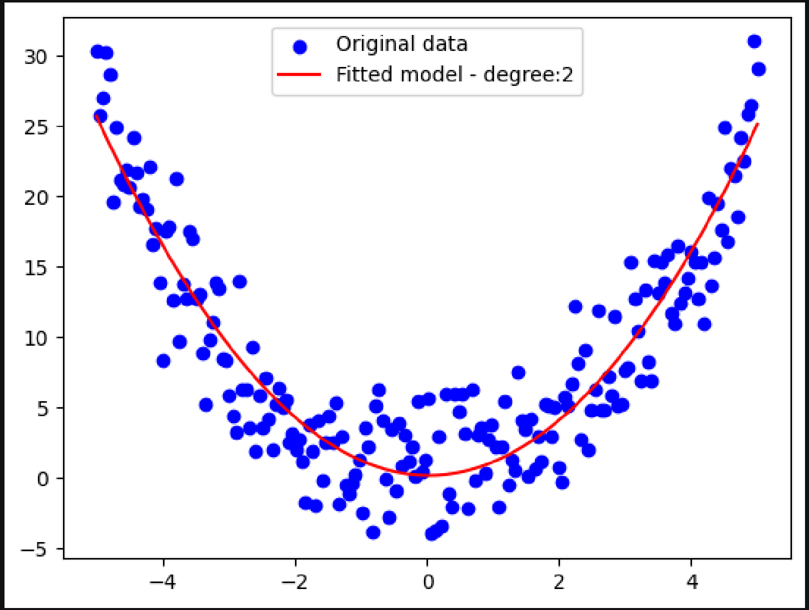


Polynomial Regression

```
Solution: Add polynomial degrees to linear regression
  - Remember: y = wx + b (Linear Model - Degree 1)
For non-linear data:
  - Degree 2 (Quadratic): y = w1x + w2x^2 + b
Examples:
  - Degree 3 (Cubic): y = w1x + w2x^2 + w3x^3 + b
  - And higher degrees...
Key Challenge:
  - Right degree = good fit
  - Too high degree = overfitting
```

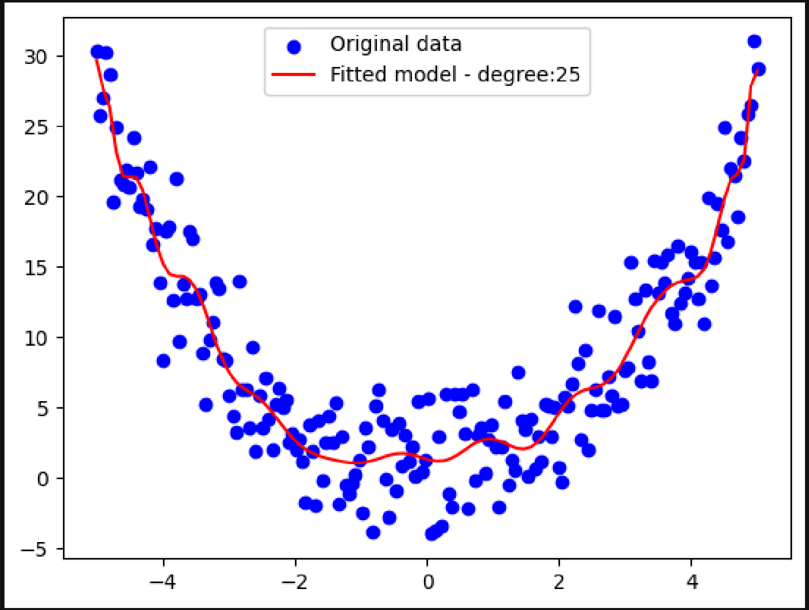
Polynomial Regression

```
Quadratic Model with Degree: 2
  - Perfect fit for the data
```



Polynomial Regression

```
Quadratic Model with Degree: 25
  - Model is Overfitting
```



Evaluation Metrics for Regression

- Mean Absolute Error (MAE):
  - Measures average prediction error
  - When to use: Best for understanding typical prediction error
  - Example Dataset: Balanced datasets without extreme outliers
- Mean Squared Error (MSE):
  - Emphasizes larger errors
  - When to use: Ideal when large errors need to be heavily penalized
  - Example Dataset: Financial predictions where large deviations are costly
- Root Mean Squared Error (RMSE):
  - Similar to MSE, but better for data with outliers
  - When to use: Use when outliers are significant in the dataset
  - Example Dataset: Environmental data with occasional extreme values
- R^2 Score:
  - Indicates fit quality (range from 0 to 1)
  - When to use: For overall fit assessment and model comparison
  - Example Dataset: General-purpose

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

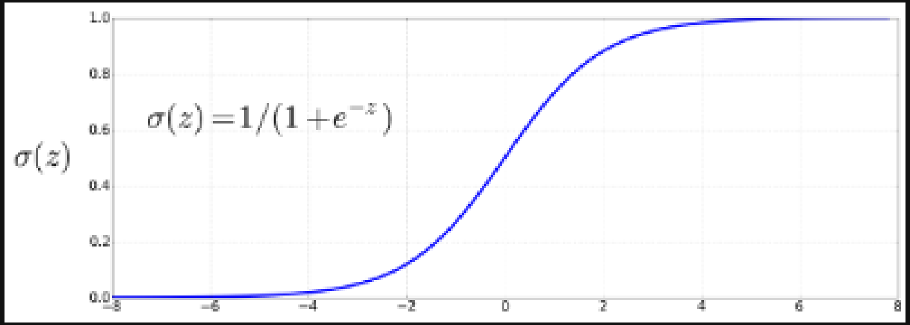
Where,

$\hat{y}$ – predicted value of y

$\bar{y}$ – mean value of y

Logistic Regression

```
- Classification Algorithm:
  - Binary Classification (0 or 1)
- Take Linear Regression Formula:
  - y = wx + b
- Plug it into the Sigmoid Function:
  - sigmoid = 1 / (1 + e^-y)
- Outputs are between 0 and 1:
  - 0.5 is the threshold
  - If sigmoid > 0.5 → 1
  - If sigmoid < 0.5 → 0
```

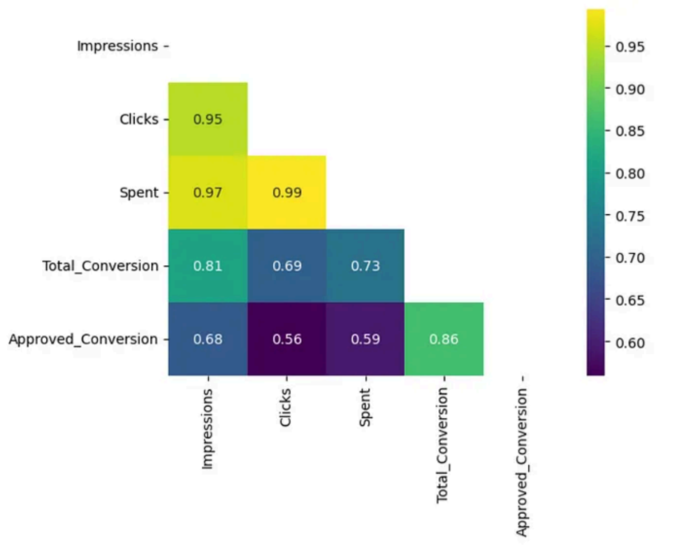


Feature Selection

```
- Feature Selection: Identifying relevant features to improve model performance
- Example: Predicting if a car should be crushed
  - Relevant: Model, Year, Miles driven
  - Irrelevant: Owner's name
- Methods:
  - Use statistical measures (e.g., Pearson Correlation)
- Approaches:
  - Filter: Statistical methods
  - Wrapper: Test different subsets
  - Embedded: Integrate into model training
```

Pearson Correlation Coefficient

```
- Pearson Correlation:
  - Measures linear relationship strength
  - Range: -1 (negative) to +1 (positive), 0 = no correlati
- Marketing Metrics Example:
  - Spent & Clicks: 0.99 (very strong)
  - Clicks & Impressions: 0.95 (strong)
  - Approved Conversion & Total Conversion: 0.86 (strong)
- Application:
  - Focus budget on metrics with strong correlations
  - Identify which features truly impact your target
```



Feature Scaling

```
- Example:
  - Wine chemical analysis dataset
  - Class based on alcohol and malic acid content
- Features are in different scales:
  - Alcohol: 11.03 - 14.83
  - Malic Acid: 0.74 - 5.80
- Features with larger scales will dominate the learning process
- By scaling features to the same range, we avoid this problem
```

Class	Alcohol	Malic
1	13.20	1.78
1	13.16	2.36
1	14.37	1.95
1	13.24	2.59
1	14.20	1.76

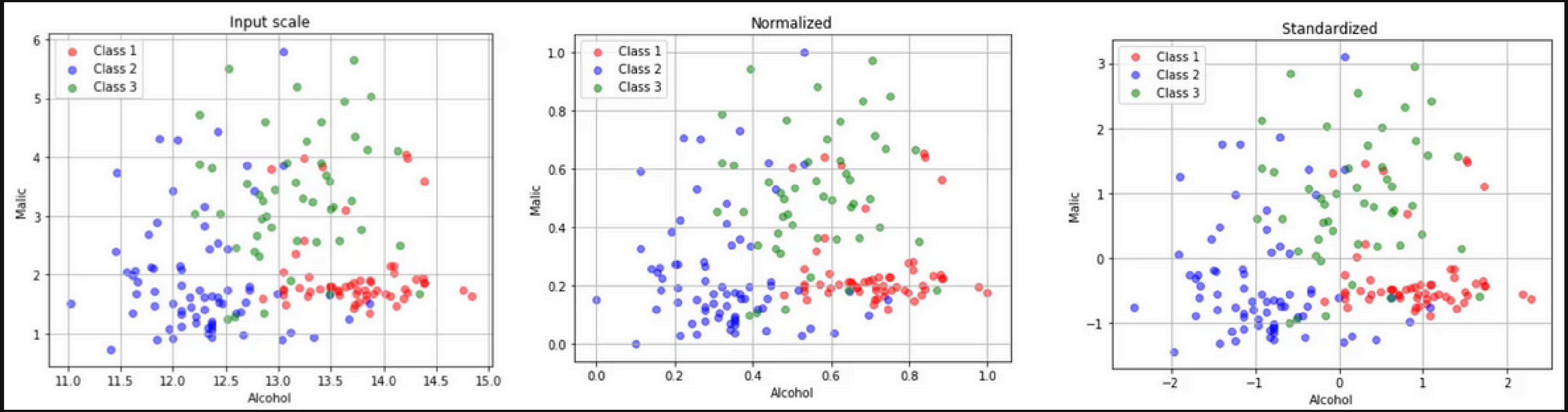
Feature Scaling - Methods

```
Normalization (Min-Max):
  - Scales features to range [0, 1]
  - Formula: x' = (x - min) / (max - min)
Standardization (Z-score):
  - Zero mean, unit variance
  - Formula: z = (x - μ) / σ
```

$$x' = \frac{x - min(x))}{max(x) - min(x)}$$

$$z = \frac{x - \mu}{\sigma}$$

Feature Scaling



Feature Scaling

```
When to use what:
  - It depends ;-)
  - Standardization:
    - Clustering, PCA
    - Distance-based algorithms
  - Normalization:
    - Neural Networks (require data on a 0-1 scale)
    - Image Processing (MNIST Dataset - scale pixel values between 0 and 1 instead of 0 and 255)
```

Let´s get our hands dirty