Class Overview

- Objective Functions
- Regularization
  - L1 Regularization
  - L2 Regularization
- Polynomial Regression
- Multi Variable Linear Regression
- Regression Model Evaluation
  - SSR
  - SST
  - R2 value

Linear/Non-Linear Relationship In Machine Learning

Relationship

Linear Relationship

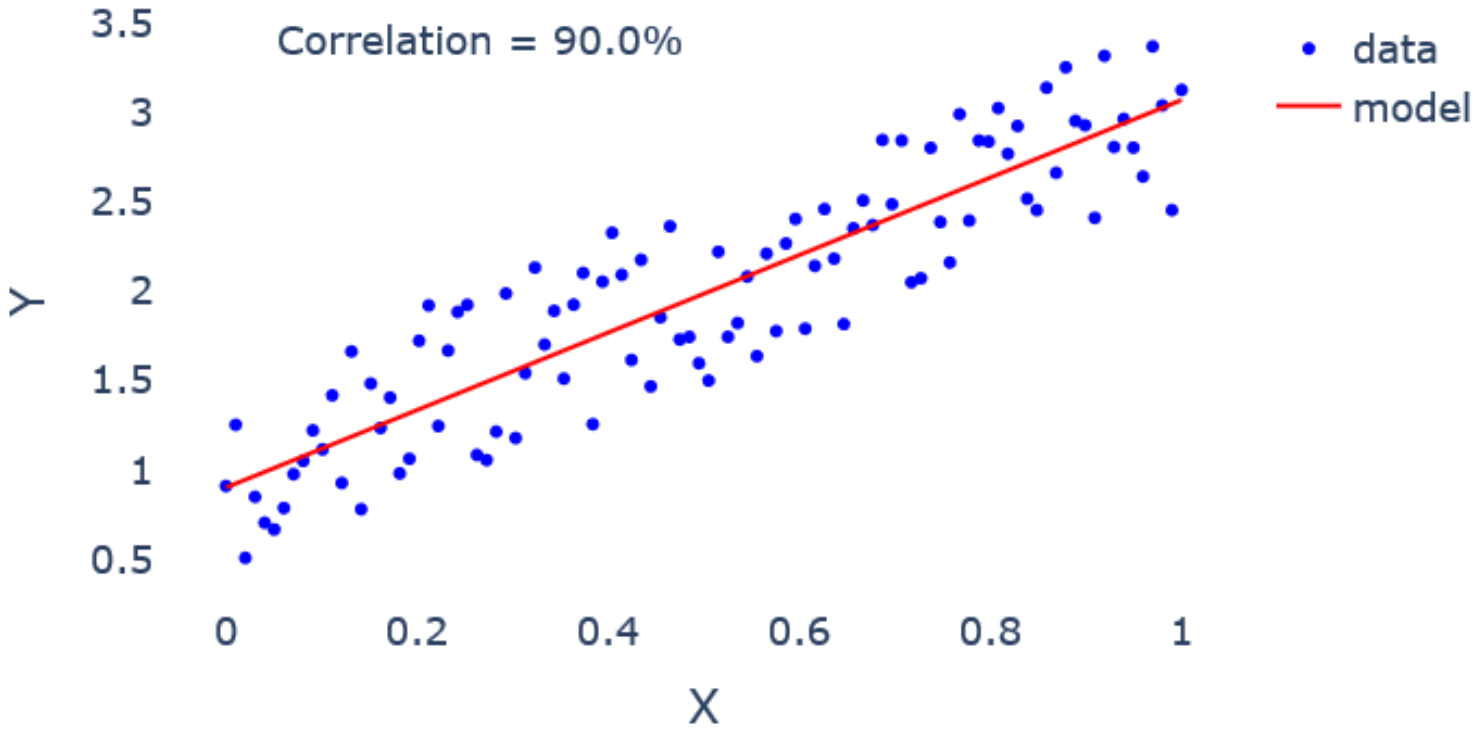


Fig: Correlation between X and Y

Relationship

Non-linear Relationship

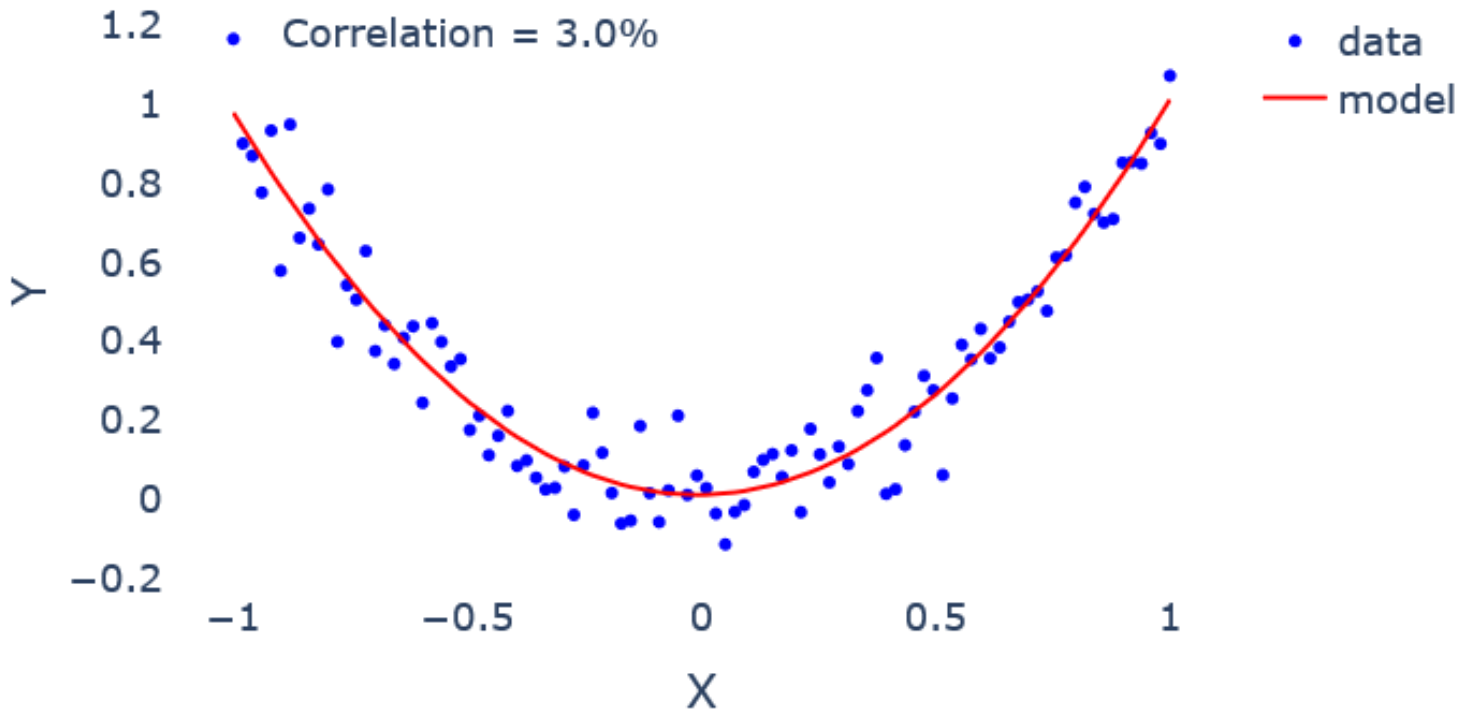
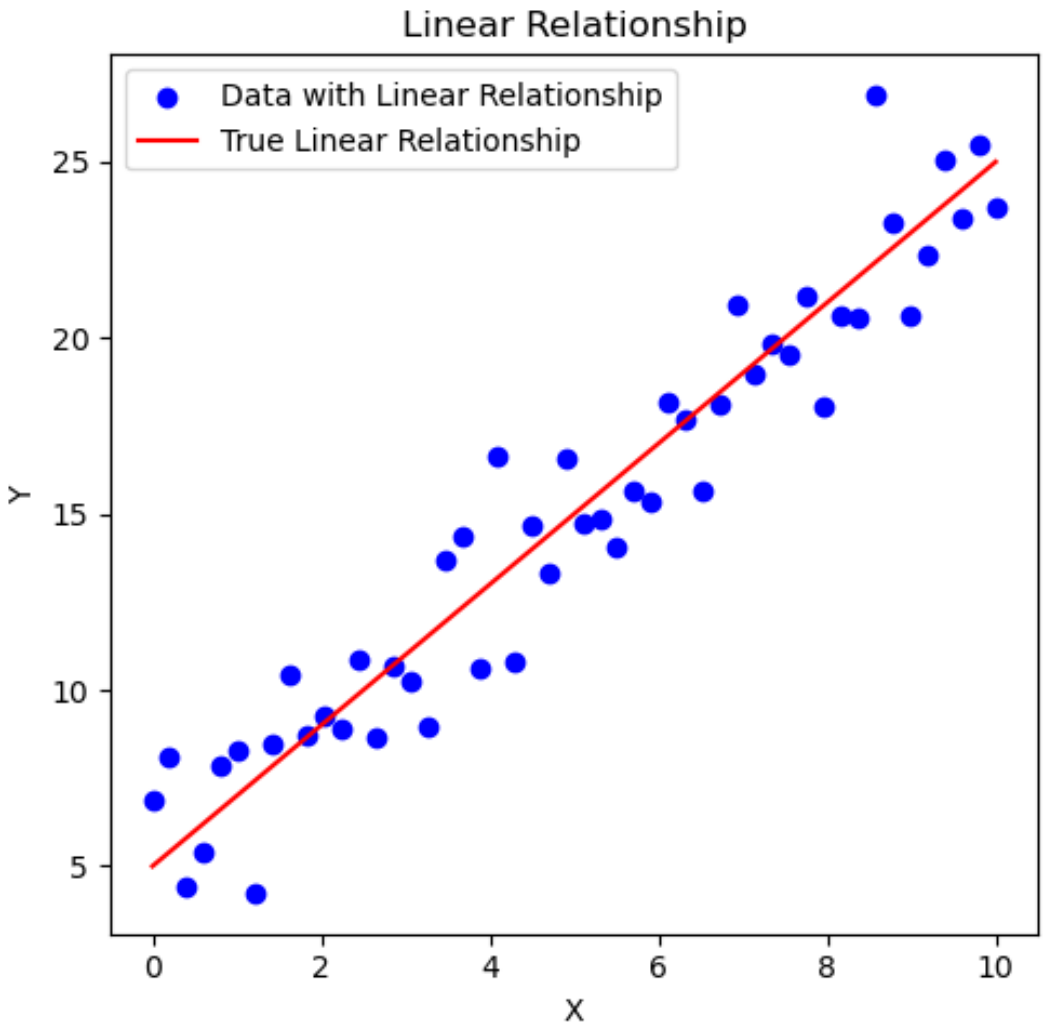
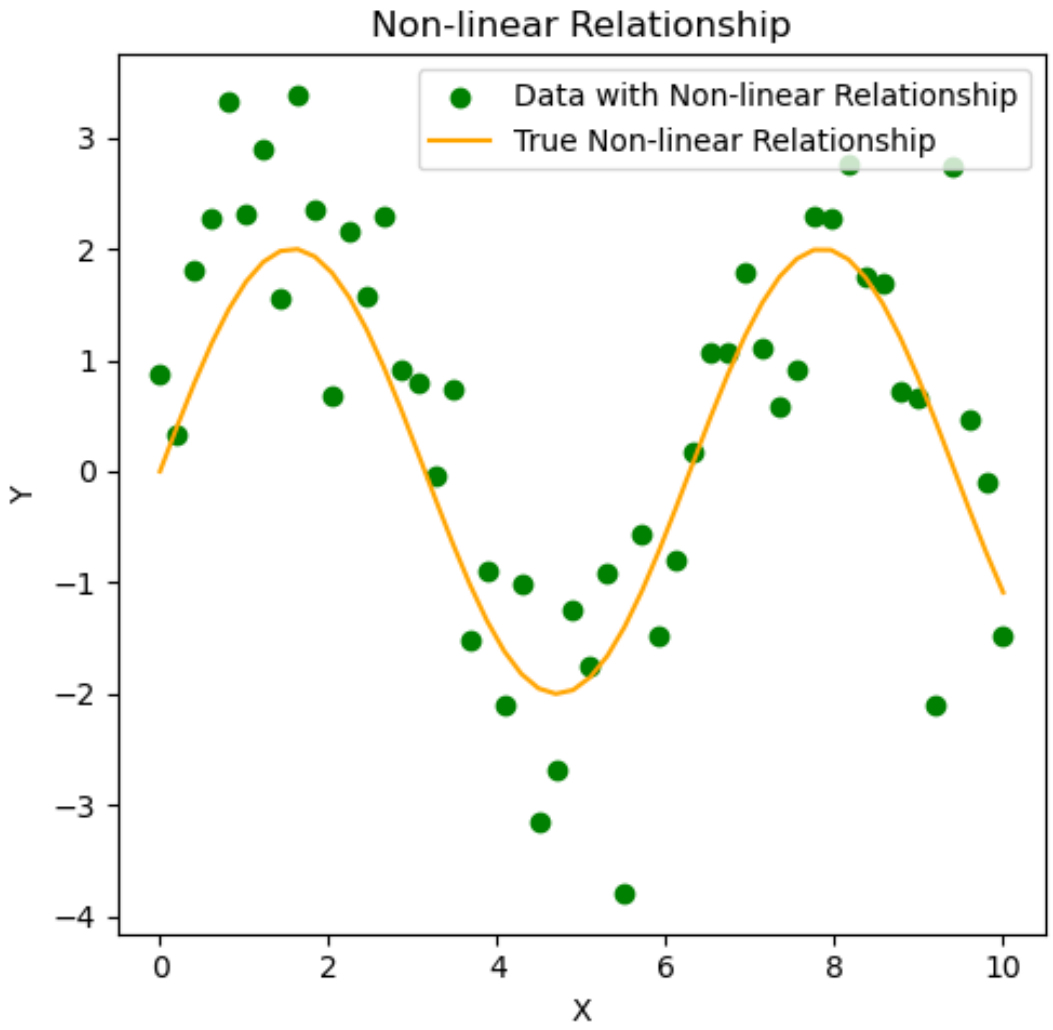


Fig: Correlation between X and Y

Relationship





Correlation

Correlation

To calculate the correlation coefficient between two variables X and Y, you can use Pearson's correlation coefficient formula.

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \cdot \sum(Y_i - \bar{Y})^2}}$$

Where:

- X_i and Y_i are individual data points for variables X and Y, respectively.
- $\bar{X}$ and $\bar{Y}$ are the means (averages) of variables X and Y, respectively.
- $\sum$ denotes the summation over all data points.

However, it's important to note that Pearson correlation measures only linear relationships and may not capture other types of relationships.

Example Mathematics:

- $X = [1, 2, 3, 4]$
- $Y = [2, 4, 6, 8]$

Step-by-Step Calculation

1. Calculate the means of X and Y:

$$\bar{X} = \frac{1+2+3+4}{4} = \frac{10}{4} = 2.5$$

$$\bar{Y} = \frac{2+4+6+8}{4} = \frac{20}{4} = 5$$

2. **Subtract the means from each data point:**

$$(X_i - \bar{X}) = [1 - 2.5, 2 - 2.5, 3 - 2.5, 4 - 2.5] = [-1.5, -0.5, 0.5, 1.5]$$

$$(Y_i - \bar{Y}) = [2 - 5, 4 - 5, 6 - 5, 8 - 5] = [-3, -1, 1, 3]$$

3. **Calculate the numerator of the correlation coefficient formula:**

$$\sum (X_i - \bar{X})(Y_i - \bar{Y}) = (-1.5 \cdot -3) + (-0.5 \cdot -1) + (0.5 \cdot 1) + (1.5 \cdot 3)$$

$$= 4.5 + 0.5 + 0.5 + 4.5 = 10$$

4. **Calculate the denominator:**

$$\sqrt{\sum(X_i - \bar{X})^2 \cdot \sum(Y_i - \bar{Y})^2}$$

$$\sum(X_i - \bar{X})^2 = (-1.5)^2 + (-0.5)^2 + (0.5)^2 + (1.5)^2 = 2.25 + 0.25 + 0.25 + 2.25 =$$

$$\sum(Y_i - \bar{Y})^2 = (-3)^2 + (-1)^2 + (1)^2 + (3)^2 = 9 + 1 + 1 + 9 = 20$$

$$\sqrt{5 \cdot 20} = \sqrt{100} = 10$$

Correlation

5. Calculate the correlation coefficient r:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \cdot \sum(Y_i - \bar{Y})^2}} = \frac{10}{10} = 1$$

So, the correlation between x and y is 1.

Correlation

To calculate the correlation coefficient between two variables X and Y, you can use Pearson's correlation coefficient formula. Another representation:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

Where:

- n is the number of data points.
- $\sum$ denotes the summation.
- x and y are the individual values of the variables.

Correlation

This will give you the correlation coefficient *r* between variables X and Y. The value of *r* will lie between -1 and 1, where:

- *r* = 1 indicates a perfect positive linear relationship,
- *r* = −1 indicates a perfect negative linear relationship,
- *r* = 0 indicates no linear relationship.

Multicollinearity Problems!

Multicollinearity Problems!

Multicollinearity is a problem in regression analysis where **two or more predictor (independent) variables are highly correlated with each other**. This means they contain **overlapping information**, which can cause several issues during model training and interpretation.

How to Detect:

- **Correlation matrix**: Look for high correlations between features.
- **Variance Inflation Factor (VIF)**: A VIF > 5 or 10 is often a red flag.

$$\text{VIF}_j = \frac{1}{1 - R_j^2}$$

VIF = 1: No multicollinearity.
VIF 1–5: Moderate correlation (often acceptable).
VIF > 5: Potential multicollinearity.
VIF > 10: Serious multicollinearity — consider fixing it.

How to Fix:

- Remove or combine highly correlated features.
- Use **regularization techniques** like **Ridge or Lasso**, which can **reduce the impact of multicollinearity**.

Multivariable Linear Regression

Multiple Variable Linear Regression:

- Multiple linear regression involves modeling the relationship between a dependent variable y and two or more independent variables $x_1, x_2, \ldots, x_k$.
- The general form of the multiple linear regression equation with k independent variables is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon$$

- Each x_i represents a different independent variable, and $\beta_0, \beta_1, \ldots, \beta_k$ are the coefficients to be estimated.

Polynomial Regression

Mathematical concepts

Multiple Variable Linear Regression:

- Multiple linear regression involves modeling the relationship between a dependent variable y and two or more independent variables $x_1, x_2, \ldots, x_k$.
- The general form of the multiple linear regression equation with k independent variables is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon$$

- Each x_i represents a different independent variable, and $\beta_0, \beta_1, \ldots, \beta_k$ are the coefficients to be estimated.

Polynomial Regression:

- Polynomial regression involves modeling the relationship between a dependent variable y and an independent variable x using a polynomial equation.
- The general form of a polynomial regression equation of degree n is:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \varepsilon$$

- Here, the independent variable x is raised to different powers up to n, allowing the model to capture non-linear relationships.

The "degree" of the polynomial refers to the **highest power of the input variable(s)** in the polynomial equation. The degree indicates the level of complexity of the model, with higher degrees allowing for more complex curves to fit the data.

- **Degree 1**:

  This represents a linear equation of the form $y = ax + b$, where a is the slope of the line and b is the y-intercept. A polynomial of degree 1 indicates a simple linear relationship between the input variable x and the output variable y.

- **Degree 2**:

  This represents a quadratic equation of the form $y = ax^2 + bx + c$. A polynomial of degree 2 can model parabolic curves, allowing for the representation of relationships where the rate of change is not constant and can accelerate or decelerate.

Polynomial Regression

Mathematical concepts

- **Degree 3**:

  This represents a cubic equation of the form $y = ax^3 + bx^2 + cx + d$. A cubic polynomial can model more complex curves that can change direction, allowing for an S-shaped curve or other non-linear patterns that a quadratic equation cannot capture.

- **Higher Degrees**: Polynomials of degree 4, 5, and beyond can model even more complex relationships with multiple inflection points and changes in direction. However, with increased degree, the polynomial model becomes more prone to overfitting, especially if the data does not support such complexity or if there is not enough data.

The formula to calculate the total number of features after applying polynomial features of degree *d* to an original set of *n* features is given by:

$$\text{Total features} = \binom{n+d}{d} = \frac{(n+d)!}{n!d!}$$

Where $\binom{n+d}{d}$ is a binomial coefficient representing the number of ways to choose d items from $n+d$ items without regard to the order, n is the original number of features, and d is the degree of the polynomial.

- For degree 2 (including original features, squares, and pairwise interactions):

$$\text{Total features} = \binom{3+2}{2} = \frac{(3+2)!}{3!2!}$$

- For degree 3 (including original features, squares, cubes, and all interactions up to third degree):

$$\text{Total features} = \binom{3+3}{3} = \frac{(3+3)!}{3!3!}$$

After performing polynomial regression:

- ❖ With a degree of 2, the feature count will increase to 10.
- ❖ With a degree of 3, the feature count will increase to 20.

Polynomial Regression

Pros and Cons

The "best" degree for polynomial regression depends on your specific dataset and the **complexity** of the **underlying relationship** that you're trying to **model between the features and the target variable**. Here are some considerations to help decide between a degree of 2 and a degree of 3, or any other degree:

Degree 2 (Quadratic):

- **Pros:**
  - Captures relationships that are not merely linear, such as **parabolic trends**.
  - Increases model complexity moderately, which can improve fitting for slightly more complex patterns without going too far into overfitting.

- **Cons:**
  - May still be **too simple** for some datasets, failing to capture higher-order relationships.

Degree 3 (Cubic):

- **Pros:**
  - Can model **more complex** patterns than quadratic, including S-shaped curves, which can be more representative of certain natural or economic phenomena.
  - Useful for datasets where the relationship between **features and target changes direction more than once.**

- **Cons:**
  - Increases the risk of overfitting, especially if the dataset is not very large, as the model becomes significantly more complex.
  - Requires more data to train effectively compared to quadratic models.

Overfitting Vs. Underfitting In Machine Learning

Overfitting and Underfitting

Overview

Training Performance: Low (Gorib)
Testing Performance: Low (Gorib)

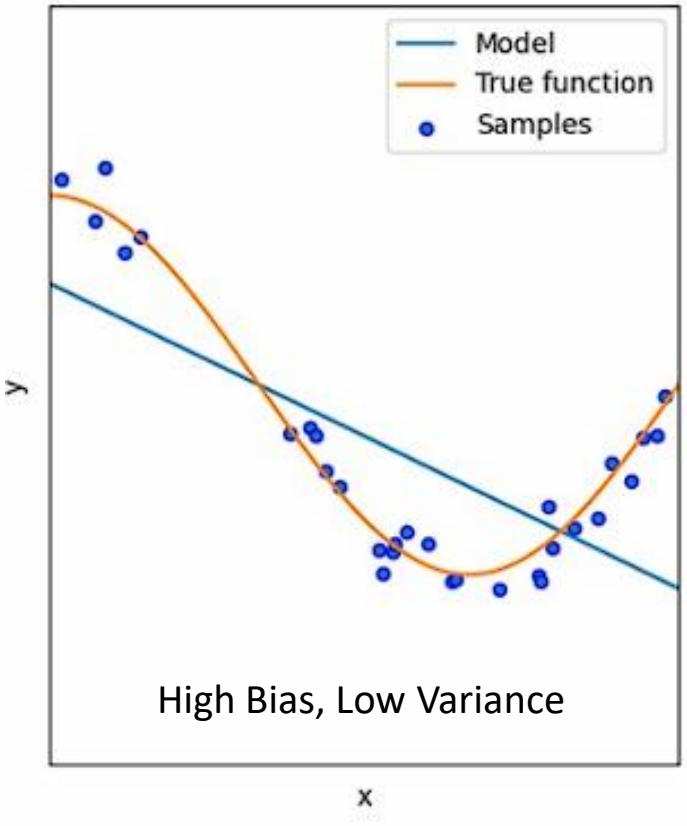


Fig 01: Underfitting

Training Performance: Good
Testing Performance: Good

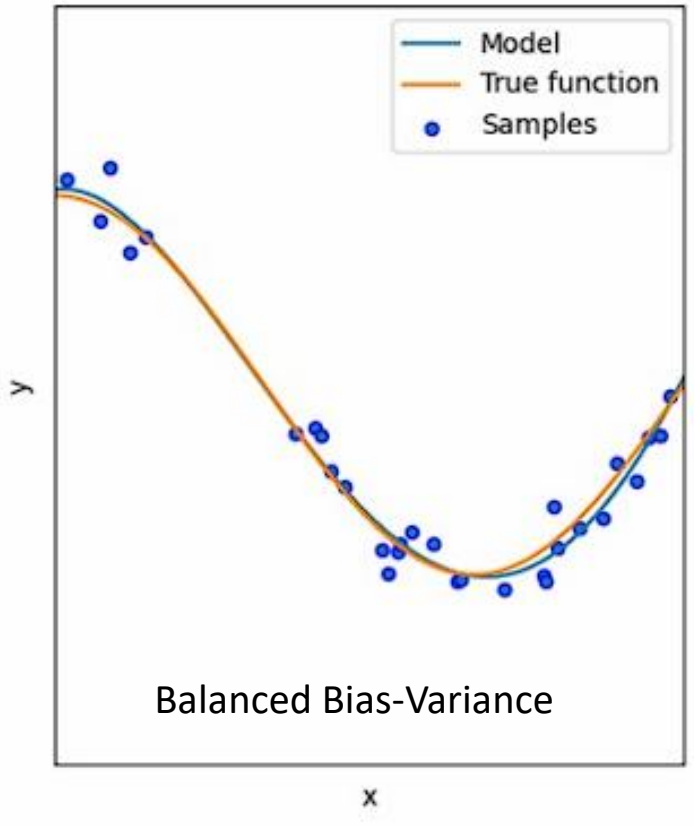


Fig 02: Best fitting

Training Performance: High (Rich)
Testing Performance: Low (Gorib)

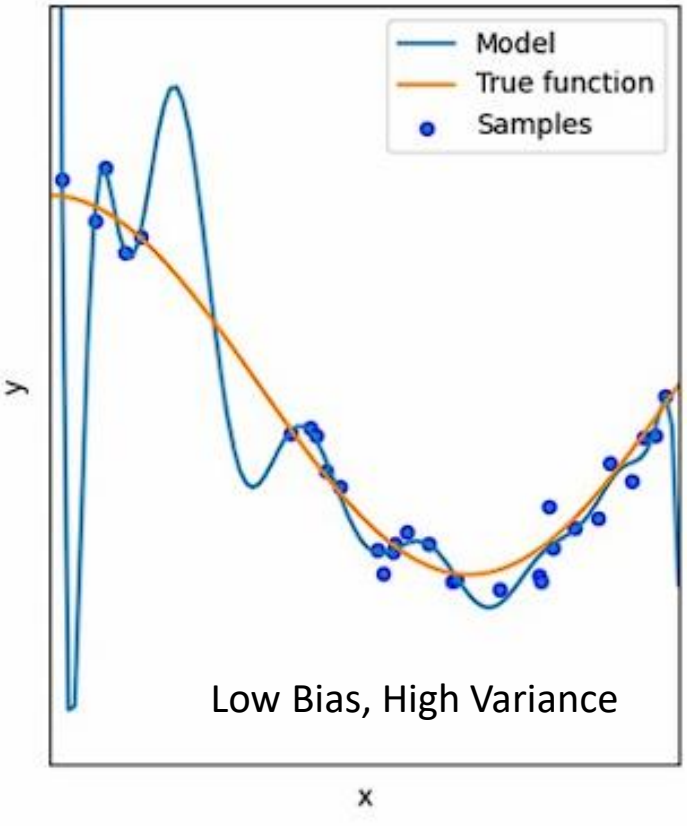


Fig 03: Overfitting

Polynomial Regression

General Considerations:

- **Model Complexity vs. Overfitting**: Higher degrees can capture more complex relationships but also increase the risk of overfitting. Overfitting happens when the model learns the noise in the training data instead of the actual underlying pattern, leading to poor performance on unseen data.

- **Cross-Validation**: To find the best degree for your polynomial regression, you can use cross-validation. This involves training the model with different degrees and evaluating their performance on a validation set or through k-fold cross-validation.

- **Domain Knowledge**: Sometimes, domain knowledge can guide the choice of degree. If you know the underlying relationship should have a specific shape or form, you can choose the degree accordingly.

High-Dimensional Statistics for Machine Learning

What is High-Dimensional Data?

Machine Learning

High-Dimensional Statistics is a field of statistics that deals with data where the **number of variables** (features or parameters) is **very large**, often comparable to or larger than the number of observations.

Mathematically, **Features >> Samples**

In classical statistics, you usually have more observations than variables, for example, 1000 patients but only 5 measurements per patient. High-dimensional statistics flips this: you might have 1000 measurements for only 50 patients. This happens a lot in modern applications like:

- Genomics (thousands of genes, few samples)
- Finance (many assets, short periods)
- Machine Learning (massive feature spaces from text, images, etc.)

High-Dimensional Statistics is a field of statistics that deals with data where the **number of variables** (features or parameters) is **very large**, often comparable to or larger than the number of observations.

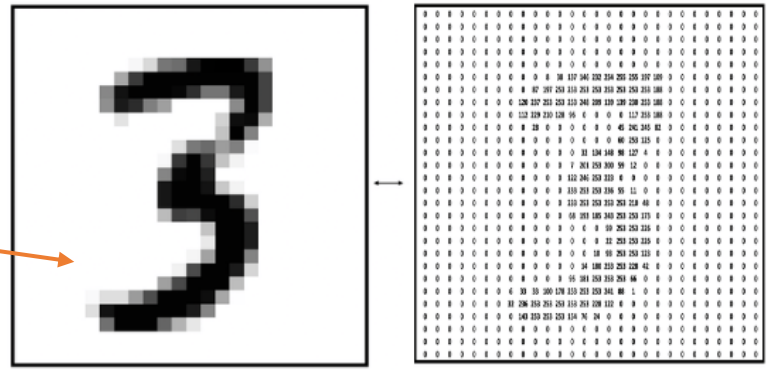
Key challenges in high-dimensional statistics:

- **Overfitting**: With so many variables, it's easy to fit random noise perfectly.
- **Sparsity**: Often, only a small subset of variables is truly important.
- **Computational complexity**: Many traditional methods become too slow or unstable.
- **Interpretability**: Harder to explain results with so many variables.

High-Dimensional Statistics is a field of statistics that deals with data where the **number of variables** (features or parameters) is **very large**, often comparable to or larger than the number of observations.

	text
0	Eddard Stark is a king in the north.
1	A king but one king : kings are everywhere.
2	Hodor was different : he was not a king .
3	But the North could not change without him.

sparse



	king	was	the	not	But	him	one	north	kings	is	in	he	Eddard	everywhere	different	could	change	but	are	Stark	North	Hodor	without
0	1	0	1	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0
1	2	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	0
2	1	2	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0
3	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1

High-Dimensional Statistics is a field of statistics that deals with data where the **number of variables** (features or parameters) is **very large**, often comparable to or larger than the number of observations.

Important concepts:

- **Regularization**: Adding penalties (like Lasso, Ridge) to control model complexity.
- **Dimensionality reduction**: Methods like PCA (Principal Component Analysis) reduce the number of variables.
- **Sparse modelling**: If only a few features matter and trying to find them.
- **Concentration of measure**: In high dimensions, random variables behave differently (e.g., distances between points become almost the same).

What is Regularization?

Regularization is a technique used in machine learning to prevent overfitting and improve the generalization of a model. Overfitting occurs when a model learns the training data too well, including its noise and outliers, to the extent that it performs poorly on new, unseen data.

Regularization methods add a penalty term to the model's loss function, discouraging the learning algorithm from assigning excessive importance to any one feature or allowing the model to become too complex.

- L1 Regularization (Lasso)
- L2 Regularization (Ridge)

Regularization (L1 & L2)

L1 Regularization(LASSO):

Lasso (Least Absolute Shrinkage and Selection Operator) adds a penalty equal to the **absolute value** of the coefficients (L1 norm) to the loss function:

$$\text{Loss} = \text{MSE (or other loss)} + \lambda \sum |w_i|$$

Regularization

Lasso / L1 with Linear Regression

L1 regularization, also known as Lasso (Least Absolute Shrinkage and Selection Operator) regularization, has the property of inducing sparsity in models. When you apply L1 regularization to linear regression (or other linear models), it adds a penalty term to the cost function that is proportional to the absolute values of the coefficients. This penalty encourages the optimization process to drive some of the coefficients to exactly zero. The L1 regularization term is typically added to the mean squared error (MSE) cost function for linear regression, resulting in the Lasso cost function::

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{m} \sum_{j=1}^{n} |\theta_j|$$

Here:

- $J(\theta)$ is the cost function.
- m is the number of training examples.
- $h_\theta(x^{(i)})$ is the predicted output for the i-th training example.
- $y^{(i)}$ is the true output for the i-th training example.
- θ_j are the coefficients (weights) of the linear regression model.
- λ is the regularization parameter, controlling the strength of the L1 penalty.
- n is the number of features.

L2 Regularization(RIDGE):

- In L2 regularization, a penalty is added to the loss function that is proportional to the square of the model parameters. The regularization term is the sum of the squared values of the weights, multiplied by a regularization parameter (lambda or alpha).
- The L2 regularization term is expressed as: $\lambda \sum_{i=1}^{n} w_i^2$, where λ is the regularization parameter and w_i are the model parameters.
- L2 regularization tends to produce weight vectors with small, non-zero values. It is effective at preventing any one feature from having an extremely large influence on the model.

Regularization

Ridge / L2 with Linear Regression

Linear regression with an L2 regularization term is often referred to as Ridge regression. The L2 regularization term adds the sum of the squared values of the coefficients to the linear regression cost function. The complete cost function for Ridge regression is given by:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Here:

- $J(\theta)$ is the cost function.
- m is the number of training examples.
- $h_\theta(x^{(i)})$ is the predicted output for the i-th training example.
- $y^{(i)}$ is the true output for the i-th training example.
- θ_j are the coefficients (weights) of the linear regression model.
- λ is the regularization parameter, controlling the strength of the L2 penalty.
- n is the number of features.

L1 Vs. L2 Regularization

- Lasso (L1 regularization) is commonly solved using **coordinate descent**, an optimization method that updates one coefficient at a time while keeping the others fixed. This works well for Lasso because the L1 penalty introduces a sharp corner at zero, making standard gradient descent difficult to apply. Coordinate descent efficiently handles this by applying soft thresholding, which can shrink some coefficients exactly to zero, making Lasso especially useful for feature selection in high-dimensional datasets.

- Ridge (L2 regularization), on the other hand, uses a **squared penalty** that is smooth and differentiable everywhere. Because of this, it does not require coordinate descent and is often solved using **closed-form solutions** (like the normal equation) or **standard gradient-based methods**. Ridge shrinks all coefficients but rarely sets any to zero, making it better suited for situations where all features are expected to contribute to the prediction. Mathematically,

$$w = (X^T X + \lambda I)^{-1} X^T y$$

Objective Functions in Machine Learning & AI

The objective function, or the cost function or loss function, is a mathematical function that a machine learning algorithm seeks to minimize during the training process. It represents a measure of the error or discrepancy between the predicted values of the model and the actual target values in the training data.

The primary goal of a machine learning algorithm is to learn a set of parameters that minimizes this objective function, thereby improving the model's ability to make accurate predictions on new, unseen data.

The objective function guides the optimization algorithm, which iteratively adjusts the model parameters to find the optimal values. or loss function is a mathematical function that a machine learning algorithm seeks to minimize during training loss function, is a mathematical function that a machine learning algorithm seeks to minimize during training loss function, is a mathematical function that a machine learning algorithm seeks to minimize during training

Objective Function

Types / Situations

- **Regression Problems:** The mean squared error (MSE) is a common objective function. It measures the average squared difference between the predicted and actual values.

- **Classification Problems:** Cross-entropy loss is often used. It quantifies the dissimilarity between the true class labels and the predicted probabilities.

Objective Function

The Mean Squared Error

The Mean Squared Error (MSE) is a commonly used objective function for regression problems. It quantifies the average squared difference between the predicted values of a model and the actual target values in the training data. The formula for MSE is as follows:

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

Here:

- n is the number of data points in the training set.
- y_i is the actual target value for the i-th data point.
- $\hat{y}_i$ is the predicted value for the i-th data point.

The MSE formula calculates the squared difference between each predicted value and its corresponding actual value, sums up these squared differences across all data points, and then divides by the number of data points (n) to obtain the average squared difference.

MSE Characteristics:

- MSE is often used when you want to penalize **larger errors** more heavily, which might be appropriate in certain contexts where larger errors are considered more critical.

- It is also commonly used in situations where the assumption of normally distributed errors holds, as minimizing MSE is equivalent to maximum likelihood estimation under this assumption. Maximizing the likelihood also provides a measure of how well the model fits the observed data. A higher likelihood value indicates a better fit between the model and the data.

- However, MSE is **sensitive to outliers** due to the squaring of errors, so if your data contains outliers, it might not be the best choice.

Objective Function

The Mean Absolute Error

The Mean Absolute Error (MAE) is a metric used to evaluate the performance of a regression model. It measures the average absolute difference between the actual and predicted values of a set of data points.

Mathematically, the Mean Absolute Error is calculated as follows:

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

Where:

- n is the number of data points.
- y_i is the actual value of the dependent variable for data point i.
- $\hat{y}_i$ is the predicted value of the dependent variable for data point i.
- $|\cdot|$ denotes the absolute value.

Objective Function

The Root Mean Squared Error

The Root Mean Squared Error (RMSE) is another common metric used to evaluate the performance of regression models, like Mean Absolute Error (MAE). However, RMSE gives more weight to larger errors because it squares the differences between predicted and actual values before taking the square root of the average.

Mathematically, the Root Mean Squared Error is calculated as follows:

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

Where:

- n is the number of data points.
- y_i is the actual value of the dependent variable for data point i.
- $\hat{y}_i$ is the predicted value of the dependent variable for data point i.

MAE Characteristics:

- MAE is preferred when you want a robust metric **to handle outliers** since it treats all errors equally.
- It is easier to interpret than MSE because it is in the same units as the original data.
- MAE might be preferred in situations where interpretability is crucial or when the distribution of errors is **not assumed to be normal**.

RMSE Characteristics:

- RMSE combines the advantages of MSE (sensitivity to larger errors) with the **interpretability** of the original scale of the data (like MAE).
- It is commonly used when you want a metric that is both interpretable and penalizes larger errors.
- RMSE is useful when the errors are **normally distributed**, and you want a metric that aligns with maximum likelihood estimation.

Regression Model Evaluation

Regression Performances

The coefficient of determination

R2 is a statistical measure that represents the proportion of the variance in the dependent variable that is predictable from the independent variable(s) in a regression model. Mathematically, it is defined as follows:

R-squared (R^2):

- Calculate R^2 using the formula: $R^2 = 1 - \frac{SSR}{SST}$.
- R^2 will be between 0 and 1.

Here,

m is the number of data points.

y_i is the actual (observed) value of the dependent variable for the i-th data point.

$\hat{y}_i$ is the predicted value of the dependent variable for the i-th data point according to the regression model.

$\bar{y}$ is the mean of the observed values of the dependent variable.

Residual Sum of Squares (SSR):

- Use your regression model to predict $\hat{y}_i$ for each data point.
- For each y_i, calculate the squared difference between the observed and predicted values: $(y_i - \hat{y}_i)^2$.
- Sum all these squared differences. This sum represents the unexplained variability and is called the Residual Sum of Squares (SSR).

Total Sum of Squares (SST):

- Calculate the mean of the observed Y values, denoted as $\bar{y}$.
- For each y_i, calculate the squared difference from the mean: $(y_i - \bar{y})^2$.
- Sum all these squared differences. This sum represents the total variability in Y and is called the Total Sum of Squares (SST).

Let’s do it with Python