Parametric vs Non-Parametric ML Algorithms

Parametric Algorithms

- Parametric algorithms make certain assumptions about the underlying data distribution or relationship between features and target variables. These algorithms have a fixed number of parameters that need to be learned from the training data. Once these parameters are learned, the model structure is determined, and the training data is no longer needed.
- Parametric algorithms are generally computationally efficient, especially when dealing with large datasets, but they might not capture complex relationships in the data well if the underlying assumptions do not hold.

Non-Parametric Algorithms

- Non-parametric algorithms make fewer assumptions about the underlying data distribution. These algorithms do not have a fixed number of parameters and can adjust their model complexity based on the amount of training data. They can capture more complex relationships but might require a larger amount of data to generalize effectively.
- Non-parametric algorithms can better capture intricate patterns in the data, but they might be computationally expensive and prone to overfitting if not properly regularized.

In summary, the choice between parametric and non-parametric algorithms depends on the nature of the data, the complexity of the underlying relationships, the amount of available data, and computational considerations. Parametric models are simpler and more interpretable but might not capture complex relationships well. Non-parametric models can capture complexity but require more data and can be computationally intensive.

Regression vs Classification

Regression is a type of supervised learning task where the goal is to predict a continuous numerical value or quantity. In other words, the output of a regression model is a real number that can be any value within a certain range. The main objective is to model the relationship between the input features (also known as independent variables or predictors) and the continuous target variable.

Examples of regression tasks include:

- Predicting house prices based on features like square footage, number of bedrooms, etc.
- Forecasting stock prices over time.
- Estimating a person's age based on various biometric measurements.

Common regression algorithms include linear regression, polynomial regression, support vector regression, and neural networks.

Classification is another type of supervised learning task where the goal is to assign input data to a specific category or class from a predefined set of classes. The output of a classification model is a discrete class label, indicating which category the input data belongs to.

Examples of classification tasks include:

- Email spam detection (categorizing emails as spam or not spam).
- Image classification (identifying whether an image contains a cat or a dog).
- Medical diagnosis (determining whether a patient has a certain disease based on symptoms).

Common classification algorithms include decision trees, random forests, support vector machines, k-nearest neighbors, and deep learning models like convolutional neural networks (CNNs) for image classification.

Supervised Learning

Supervised learning is a type of machine learning where the algorithm learns from labeled training data. In this approach, the training data consists of input features and their corresponding target labels. The algorithm's goal is to learn a mapping between the input features and the target labels so that it can make accurate predictions on new, unseen data.

Examples of supervised learning tasks include:

- Regression: Predicting a continuous numerical value (e.g., predicting house prices).
- Classification: Assigning inputs to predefined categories (e.g., email spam detection, image classification).

In supervised learning, the algorithm is "supervised" by having access to the correct answers during training, which enables it to learn patterns and relationships in the data.

Unsupervised Learning

Unsupervised learning involves training an algorithm on unlabeled data, where the algorithm's objective is to find patterns or structures in the data without explicit target labels. The algorithm seeks to discover inherent relationships, groupings, or distributions within the data.

Examples of unsupervised learning tasks include:

- Clustering: Grouping similar data points together (e.g., customer segmentation based on purchasing behavior).
- Dimensionality Reduction: Reducing the number of input features while retaining meaningful information (e.g., Principal Component Analysis).
- Anomaly Detection: Identifying unusual or rare instances in the data.

Unsupervised learning is used when there is no predefined notion of correct answers, and the algorithm explores the data's intrinsic properties to create meaningful representations.

Semi-Supervised Learning

Semi-supervised learning is a hybrid approach that combines elements of both supervised and unsupervised learning. In this scenario, the training data contains a mix of labeled and unlabeled examples. The algorithm uses the labeled data to learn patterns and relationships just like in supervised learning, but it also leverages the unlabeled data to improve its understanding of the data distribution and uncover additional patterns.

Semi-supervised learning is especially useful when obtaining large amounts of labeled data is expensive or time-consuming. By using a small amount of labeled data in combination with a larger amount of unlabeled data, the algorithm can potentially achieve better performance than pure supervised learning.

In summary, supervised learning involves learning from labeled data, unsupervised learning focuses on discovering patterns in unlabeled data, and semi-supervised learning combines both labeled and unlabeled data to improve learning outcomes. The choice of which approach to use depends on the problem at hand, the availability of labeled data, and the desired outcomes.

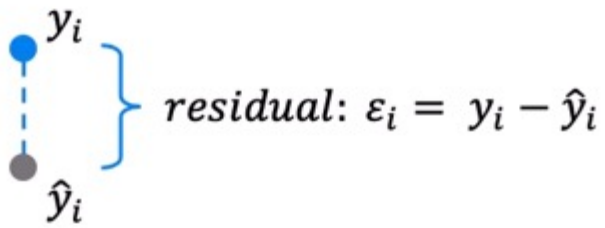
Linear Regression

Linear regression is a fundamental statistical and machine learning technique used for modeling the relationship between a dependent variable (also called the target or response variable) and one or more independent variables (also called predictors or features). It assumes a linear relationship between the independent variables and the dependent variable. The goal of linear regression is to find the best-fitting linear equation that describes this relationship.

A **causal relationship** is a connection between two or more variables where changes in one variable directly cause changes in another variable. In other words, a causal relationship implies that changes in the independent variable lead to changes in the dependent variable, and there is a cause-and-effect mechanism at play.

Simple Linear Regression-

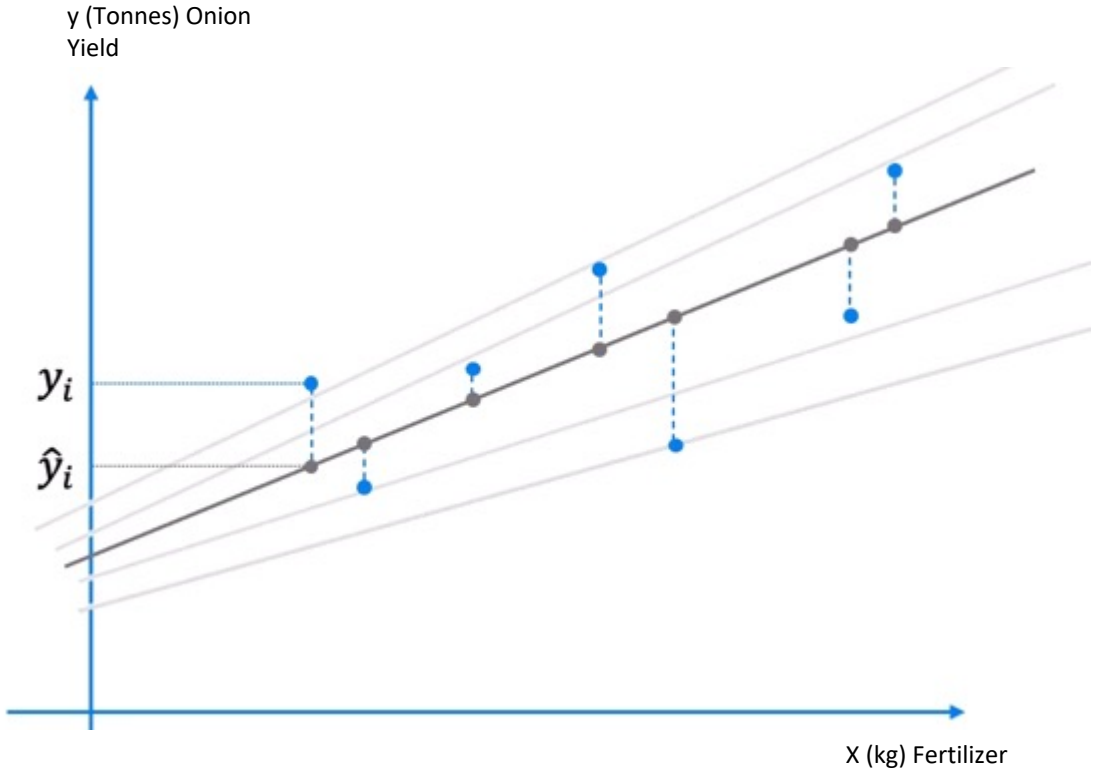
Ordinary Least Squares:



$\hat{y} = b_0 + b_1 X_1$

b_0, b_1 such that:

$SUM(y_i - \hat{y}_i)^2$ is minimized



Least Squares stands for minimum squares error or SSE

Least Square Linear Equation

The least squares linear equation is the result of fitting a linear model to data in such a way that it minimizes the sum of the squared differences between the observed values and the values predicted by the linear equation. This method is known as least squares regression and is commonly used to find the best-fitting line through a set of data points.

The equation of a straight line in its general form is:

$$y = mx + b$$

Where:

- y is the dependent variable (the variable you're trying to predict),
- x is the independent variable (the variable you're using to make predictions),
- m is the slope of the line,
- b is the y-intercept (the value of y when x is 0).

In the context of least squares regression, the goal is to find the values of m and b that minimize the sum of the squared differences between the observed y values and the predicted y values (based on the linear equation). Mathematically, the least squares method aims to minimize the following sum:

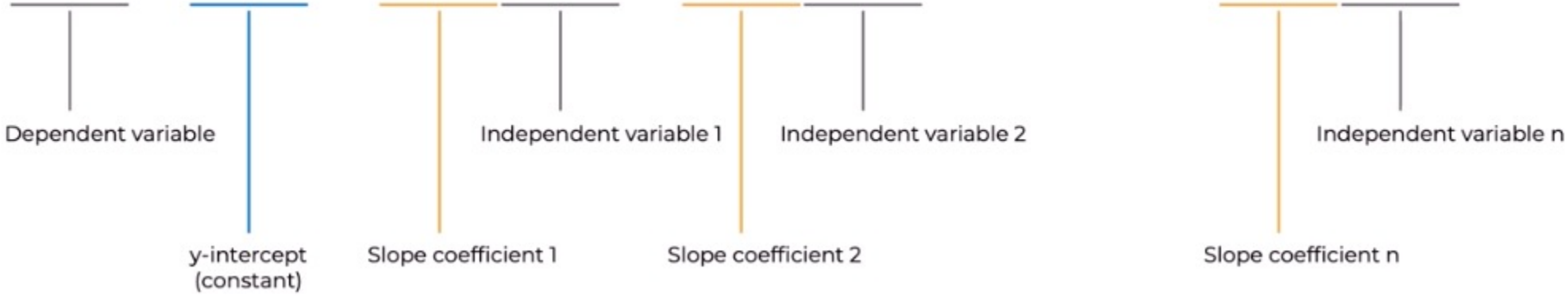
$$\sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

Where n is the number of data points, (x_i, y_i) are individual data points, and $(mx_i + b)$ is the predicted y value for a given x_i.

The values of m and b that minimize the sum of squared differences can be calculated using various methods, including calculus and matrix algebra. The result is the equation of the least squares regression line that best fits the data.

Multiple Linear Regression

$$\hat{y} = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n$$

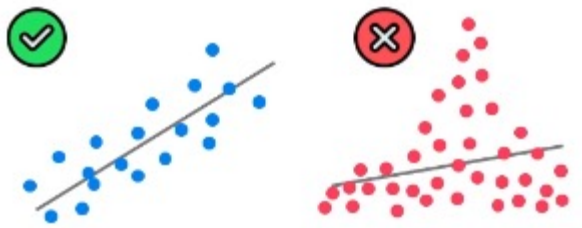


$$\text{Onion}\,[t] = 8t + 3\,\frac{t}{kg} \times Fertilizer[kg] - 0.54\,\frac{t}{°C} \times AvgTemp[°C] + 0.04\,\frac{t}{mm} \times Rain[mm]$$

Assumptions of Linear Regression

Linearity-linear relationship between Y and each X

Linear regression needs the relationship between the independent and dependent variables to be linear. It is also important to check for outliers since linear regression is sensitive to outlier effects. The linearity assumption can best be tested with scatter plots, the following two examples depict two cases, where no and little linearity is present.

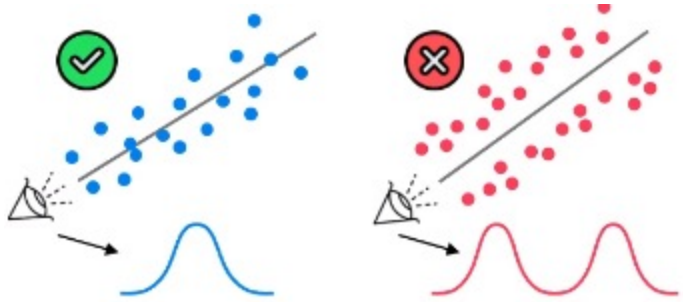


Fixes for Linearity:

- Run a non-linear regression -Classification and Regression Trees, Naive Bayes, K-Nearest Neighbors, Learning Vector Quantization and Support Vector Machines.
- Transform the relationship
  - Exponential transformation
  - Logarithmic transformation

Multivariate Normality-

linear regression analysis requires that the errors between observed and predicted values (i.e., the residuals of the regression) should be normally distributed. This assumption may be checked by looking at a histogram



Fixes

- Central limit theorem

Assumptions of Linear Regression

No Multicollinearity-

It assumes that the independent variables are not highly correlated with each other.

Multicollinearity may be checked multiple ways:

1) Correlation matrix – When computing a matrix of Pearson's bivariate correlations among all independent variables, the magnitude of the correlation coefficients should be less than .80.

2) Variance Inflation Factor (VIF) – The VIFs of the linear regression indicate the degree that the variances in the regression estimates are increased due to multicollinearity. VIF values higher than 5 or 10 indicate that multicollinearity is a problem.

If multicollinearity is found in the data, one possible solution is to center the data. To center the data, subtract the mean score from each observation for each independent variable. However, the simplest solution is to identify the variables causing multicollinearity issues (i.e., through correlations or VIF values) and removing those variables from the regression.

No Endogeneity-

Endogeneity refers to a situation in which there is a correlation between the independent variables and the error term in a regression model. This can lead to biased and inconsistent coefficient estimates, which undermines the reliability of the results. In other words, endogeneity violates the assumption that the independent variables are not affected by the error term. If they are correlated, it can lead to omitted variable bias

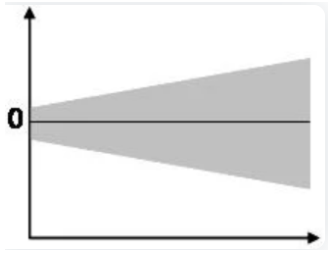
Omitted variable bias is a common issue in regression analysis that occurs when an important variable is left out of a regression model. This omission can lead to biased and unreliable coefficient estimates for the included variables, as well as incorrect conclusions about their relationships with the dependent variable. Omitted variable bias arises because the omitted variable might be correlated with both the included independent variables and the dependent variable.

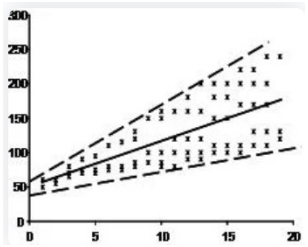
Assumptions of Linear Regression

Homoscedasticity-

This assumption states that the variance of error terms are similar across the values of the independent variables. A plot of standardized residuals versus predicted values can show whether points are equally distributed across all values of the independent variables.

A scatterplot of residuals versus predicted values is good way to check for homoscedasticity. There should be no clear pattern in the distribution; if there is a cone-shaped pattern (as shown below), the data is heteroscedastic.





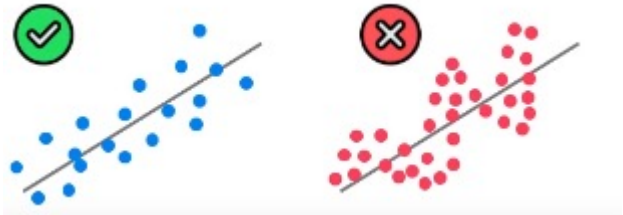
Fixes:

- Check for Omitted variable bias
- Look for outliers
- Log transformation

No Autocorrelation-

Autocorrelation is another assumption of linear regression that needs to be met. Autocorrelation occurs when the residuals are not independent from each other**.** In other words, it is a measure of similarity or correlation between adjacent data points, where data points are affected by the values of points that came before. For instance, this typically occurs in stock prices, where the price is not independent from the previous price. Ideally, model errors should be independent and identically distributed and thus should have no patterns in them**.** Autocorrelation can cause problems like invalid linear regression conclusions or interpretations.

To test the linear regression model for autocorrelation, we can use the **Durbin-Watson test**. Durbin-Watson's d tests the null hypothesis that the residuals are not linearly autocorrelated. Values around 2 indicate no autocorrelation, and as a rule of thumb, values of $1.5 < d < 2.5$ show that there is no autocorrelation in the data**.** If autocorrelation is present, we need to investigate the omission of a key predictor or use methods like autoregressive models to estimate the regression parameters of the Y versus X relationship



Math Behind Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. The goal is to find the best-fitting line that minimizes the sum of squared differences between the observed values and the values predicted by the linear equation. The mathematical foundation of linear regression involves concepts from linear algebra and calculus.

Here's a step-by-step overview of the mathematical process behind simple linear regression (one independent variable):

1. **Model Assumption:**
   The model assumes a linear relationship between the dependent variable y and the independent variable x:
   $y = mx + b + \varepsilon$
   where m is the slope, b is the intercept, and ε represents the error term (deviation from the true line).

Math Behind Linear Regression

2. **Residuals and Error:**

   The error for each data point is the difference between the observed y value and the predicted y value from the linear equation:

   $$\varepsilon = y - (mx + b)$$

3. **Objective Function - Minimization of Residuals:**

   The goal of linear regression is to find the values of m and b that minimize the sum of squared residuals (also known as the least squares criterion):

   $$\text{Minimize} \sum_{i=1}^{n} \varepsilon_i^2$$

   This can be expanded and rewritten as:

   $$\text{Minimize} \sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

4. **Minimization Using Calculus:**

   To find the values of m and b that minimize the sum of squared residuals, calculus is used. Taking partial derivatives with respect to m and b, setting them to zero, and solving the resulting equations gives the optimal values:

   $$m = \frac{n(\sum_{i-1}^{n} x_i y_i) - (\sum_{i-1}^{n} x_i)(\sum_{i-1}^{n} y_i)}{n(\sum_{i-1}^{n} x_i^2) - (\sum_{i-1}^{n} x_i)^2}$$

   $$b = \frac{(\sum_{i-1}^{n} y_i)(\sum_{i-1}^{n} x_i^2) - (\sum_{i-1}^{n} x_i y_i)(\sum_{i-1}^{n} x_i)}{n(\sum_{i-1}^{n} x_i^2) - (\sum_{i-1}^{n} x_i)^2}$$

Math Behind Linear Regression

5. **Interpretation of Coefficients:**

   The coefficient m represents the change in the dependent variable y for a unit change in the independent variable x. The intercept b represents the value of y when x is zero.

6. **Fitting the Line:**

   Using the calculated m and b, the linear regression line $y = mx + b$ is fitted to the data.

This explanation covers the basics of simple linear regression. For multiple linear regression (when there are more than one independent variable), the process involves matrix notation and vector calculus to extend the concept to higher dimensions.

Decomposition Variability

The decomposition of variability refers to the process of breaking down the total variability in a dataset or a statistical model into its constituent parts or sources. In the context of regression analysis, the variability in the dependent variable can often be decomposed into different components that help us understand the factors contributing to the observed variations.

Let's explore how variability can be decomposed in the context of regression analysis:

1. **Total Variability (Total Sum of Squares, SST):** This represents the total variation in the dependent variable. It measures the differences between the observed values of the dependent variable and their mean.

$$\text{SST} = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

Where y_i is the observed value of the dependent variable for observation i, and $\bar{y}$ is the mean of the observed values.

Explained Variability (Regression Sum of Squares, SSR): This represents the variation in the dependent variable that is explained by the regression model. It measures how well the model fits the data.

$$\text{SSR} = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$$

Where $\hat{y}_i$ is the predicted value of the dependent variable for observation i.

Unexplained Variability (Residual Sum of Squares, SSE): This represents the variation in the dependent variable that is not explained by the regression model. It captures the errors or residuals of the model.

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

Decomposition of Variability Identity: The decomposition of variability identity shows the relationship between the total variability, explained variability, and unexplained variability:

$$\text{SST} = \text{SSR} + \text{SSE}$$

In other words, the total variability can be partitioned into the variability explained by the model (regression) and the variability that remains unexplained (residuals).

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n} e_i^2$$

Decomposition Variability-AIC

The **Akaike information criterion (AIC)** is a mathematical method for evaluating how well a model fits the data it was generated from. In statistics, AIC is used to compare different possible models and determine which one is the best fit for the data. Once you've created several possible models, you can use AIC to compare them. Lower AIC scores are better, and AIC penalizes models that use more parameters. So if two models explain the same amount of variation, the one with fewer parameters will have a lower AIC score and will be the better-fit model.

How to compare models using AIC

AIC determines the relative information value of the model using the maximum likelihood estimate and the number of parameters (independent variables) in the model. The formula for AIC is:

$$AIC=2k+n\log(RSS/n) \quad (k=d+2)$$

K is the number of independent variables used and L is the log-likelihood estimate (a.k.a. the likelihood that the model could have produced your observed y-values). The default K is always 2, so if your model uses one independent variable your K will be 3, if it uses two independent variables your K will be 4, and so on.

To compare models using AIC, you need to calculate the AIC of each model. If a model is more than 2 AIC units lower than another, then it is considered significantly better than that model.

Decomposition Variability-BIC

The Bayesian Information Criterion (BIC), also known as the Schwarz criterion, is another model selection criterion that, like the Akaike Information Criterion (AIC), helps you choose the most appropriate model among a set of candidates. BIC is particularly useful in situations where you want to avoid overfitting and select a model that is not only well-fitting but also parsimonious.

The BIC formula is given by:

$$\mathrm{BIC} = -2\ln(L) + k\ln(n)$$

Where:

- L is the likelihood of the data given the model.
- k is the number of estimated parameters in the model.
- n is the sample size.

Like AIC, the BIC penalizes models for having more parameters. However, the penalty term in BIC is larger because it includes the natural logarithm of the sample size. This means that BIC tends to favor simpler models more strongly than AIC does.

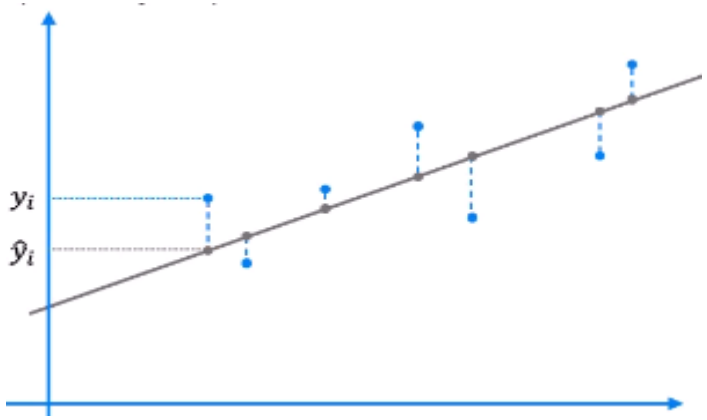
Here's how to use BIC for model selection:

1. **Fit Models:** Fit different models to your data, just like with AIC.
2. **Calculate BIC:** Calculate the BIC value for each model using the formula.
3. **Compare BIC:** Compare the BIC values of the models. The model with the lowest BIC is preferred, as it indicates a better trade-off between model fit and complexity.
4. **Interpretation:** Similar to AIC, a difference in BIC values of around 2 or more between models is generally considered meaningful. Lower BIC values indicate a better fit.
5. **Sample Size Effect:** BIC is more sensitive to sample size than AIC. As the sample size increases, the penalty for model complexity in BIC becomes more significant, encouraging the selection of simpler models.
6. **Model Complexity:** Because of the stronger penalty for complexity in BIC, it is often more conservative in model selection, favoring simpler models even more.
7. **Similarity to AIC:** BIC and AIC both aim to balance model fit and complexity but apply different penalties. The choice between them depends on the context and the trade-off you wish to make.

R-Squared

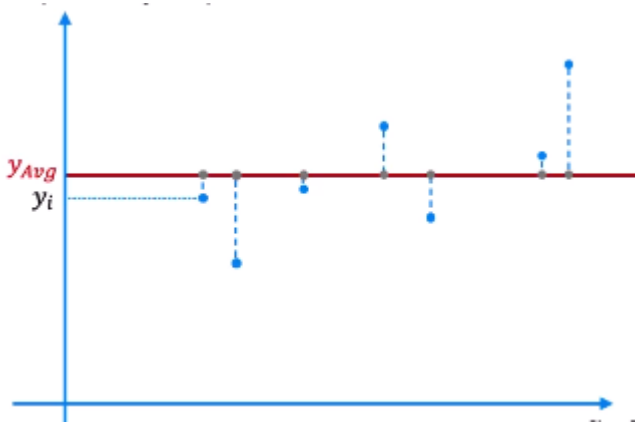
R-squared (Coefficient of Determination) is a statistical measure used to assess the proportion of the variance in the dependent variable that is explained by the independent variables in a regression model. It provides insight into how well the independent variables account for the variability observed in the dependent variable. R-squared is commonly used to evaluate the goodness of fit of a regression model.

Regression



$$SS_{res} = SUM(y_i - \hat{y}_i)^2$$

Average



$$SS_{tot} = SUM(y_i - y_{avg})^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Rule of Thumb

1.0 = Perfect fit (suspicious)
~0.9 = Very good
<0.7 = Not great
<0.4 = Terrible
<0 = Model makes no sense for this data

Adjusted R-Squared

Adjusted R-squared penalizes excessive use of variables so its value will be less than R-squared

Adjusted R-squared is a modification of the regular R-squared (coefficient of determination) that takes into account the number of independent variables in a regression model. It addresses a limitation of the standard R-squared, which tends to increase as more independent variables are added to the model, even if those variables do not significantly improve the model's explanatory power.

The formula for the adjusted R-squared is:

$$\text{Adjusted } \mathrm{R}^2 = 1 - \frac{(1-R^2)\cdot(n-1)}{n-k-1}$$

Where:

- R^2 is the regular R-squared value.
- n is the number of observations in the dataset.
- k is the number of independent variables in the model.

Feature Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units.
If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smalller values as the lower values, regardless of the unit of the values.

Why use Feature Scaling?

- Scaling guarantees that all features are on a comparable scale and have comparable ranges.
- Algorithm performance improvement: When the features are scaled, several machine learning methods, including gradient descent-based algorithms, distance-based algorithms (such k-nearest neighbours), and support vector machines, perform better or converge more quickly.
- Preventing numerical instability: Numerical instability can be prevented by avoiding significant scale disparities between features.
- Scaling features makes ensuring that each characteristic is given the same consideration during the learning process. Without scaling, bigger scale features could dominate the learning, producing skewed outcomes.

Normalization or Min-Max Scaling is used to transform features to be on a similar scale. The new point is calculated as:
`X_new = (X - X_min)/(X_max - X_min)`
This scales the range to [0, 1] or sometimes [-1, 1]

Standardization or Z-Score Normalization is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called as Z-score.
`X_new = (X - mean)/Std`
The range is not bound but usually between [-3,+3]

Feature Scaling

Normalized Data Vs Standardized Data

- Normalization is used when the data doesn't have Gaussian distribution whereas Standardization is used on data having Gaussian distribution.
- Normalization scales in a range of [0,1] or [-1,1]. Standardization is not bounded by range.
- Normalization is highly affected by outliers. Standardization is slightly affected by outliers.
- Normalization is considered when the algorithms do not make assumptions about the data distribution. Standardization is used when algorithms make assumptions about the data distribution.

If you see a bell-curve in your data then standardization is more preferable. For this, you will have to plot your data. If your dataset has extremely high or low values (outliers) then standardization is more preferred because usually, normalization will compress these values into a small range.

Feature Selection

- Univariate feature selection is a method used to select the most important features in a dataset. The idea behind this method is to evaluate each individual feature's relationship with the target variable and select the ones that have the strongest correlation.

Regularization

- Regularization is one of the ways to improve our model to work on unseen data by ignoring the less important features.
- Regularization minimizes the validation loss and tries to improve the accuracy of the model.
- It avoids overfitting by adding a penalty to the model with high variance, thereby shrinking the beta coefficients to zero.

There are two types of regularization:

1.Lasso Regularization

2.Ridge Regularization

Bias Variance Trade-off

What is Variance?

Variance tells us about the spread of the data points. It calculates how much a data point differs from its mean value and how far it is from the other points in the dataset.

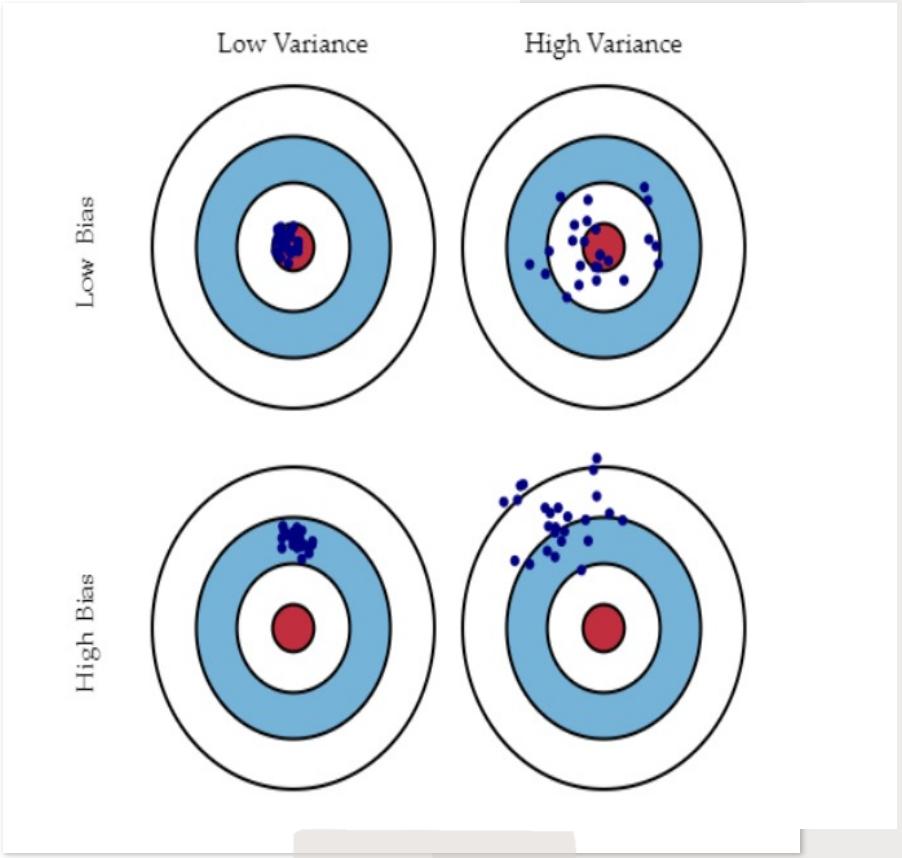
What is Bias?

It is the difference between the average prediction and the target value.

Low bias and low variance will give a balanced model, whereas high bias leads to underfitting, and high variance lead to overfitting.

- **Low Bias:** The average prediction is very close to the target value
- **High Bias:** The predictions differ too much from the actual value
- **Low Variance:** The data points are compact and do not vary much from their mean value
- **High Variance:** Scattered data points with huge variations from the mean value and other data points.

To make a good fit, we need to have a correct balance of bias and variance.



Bias Variance Trade-off

The bias-variance trade-off is a fundamental concept in machine learning and statistical modeling that deals with the relationship between a model's prediction errors due to bias and variance. It helps us understand the sources of errors in a model and guides us in finding the right level of complexity for a model.

Here's how the bias-variance trade-off works:

1. **Bias:**
    1. Bias refers to the error introduced by approximating a real-world problem, which may be complex, by a simplified model. A model with high bias makes strong assumptions about the data and may oversimplify the underlying relationships. It tends to consistently underperform on both the training and test data.
    2. High-bias models are often too simplistic and fail to capture the underlying patterns in the data. They are said to be underfitting the data.
2. **Variance:**
    1. Variance refers to the model's sensitivity to small fluctuations or noise in the training data. A model with high variance is highly flexible and can fit the training data very closely. However, it tends to perform poorly on new, unseen data because it's capturing noise rather than true patterns.
    2. High-variance models can be overly complex and fit the training data too closely. They are said to be overfitting the data.
3. **Trade-off:**
    1. The goal of a machine learning model is to find a balance between bias and variance that minimizes the overall prediction error on new data (test data).
    2. As you increase the complexity of a model (for example, by adding more features or increasing the degree of polynomial regression), the variance tends to increase while bias decreases. This can lead to overfitting.
    3. Conversely, as you decrease the complexity of a model, bias increases while variance decreases. This can lead to underfitting.

The bias-variance trade-off illustrates that there's a "sweet spot" in model complexity where the combined error due to bias and variance is minimized, resulting in better generalization to new data.

To summarize:

- **Underfitting:** High bias, low variance. The model is too simple to capture the underlying patterns in the data.
- **Overfitting:** Low bias, high variance. The model is too complex and fits noise in the data.
- **Balanced Model:** Moderate bias, moderate variance. The model generalizes well to new data.

Finding this balance often involves techniques such as regularization, cross-validation, and careful feature selection. The goal is to choose a model that is complex enough to capture essential patterns in the data but not so complex that it fits noise or idiosyncrasies.

What is Lasso Regularization (L1)?

- It stands for Least Absolute Shrinkage and Selection Operator
- It adds L1 the penalty
- L1 is the sum of the absolute value of the beta coefficients

Cost function = Loss + λ + Σ ||w||

Here,

Loss = sum of squared residual

λ = penalty

w = slope of the curve

What is Ridge Regularization (L2)

- It adds L2 as the penalty
- L2 is the sum of the square of the magnitude of beta coefficients

Cost function = Loss + λ + Σ $||w||^2$

Here,

Loss = sum of squared residual

λ = penalty

w = slope of the curve

λ is the penalty term for the model. As λ increases cost function increases, the coefficient of the equation decreases and leads to shrinkage.

Comparing Lasso and Ridge Regularization techniques

L1 Regularization	L2 Regularization
Penalty is the absolute value of coefficients	Penalty is the square of the coefficients
Estimate median of the data	Estimate mean of the data
Shrinks coefficients to zero	Shrinks coefficients equally
Can be used for dimension reduction and feature selection	Useful when we have collinear features

Difference between Fit,Transform and Fit_Transform methods

Method	Purpose	Syntax	Example
fit()	Learn and estimate the parameters of the transformation	`estimator.fit(X)`	`estimator.fit(train_data)`
transform()	Apply the learned transformation to new data	`transformed_data = estimator.transform(X)`	`transformed_data = estimator.transform(test_data)`
fit_transform()	Learn the parameters and apply the transformation to new data	`transformed_data = estimator.fit_transform(X)`	`transformed_data = estimator.fit_transform(data)`

Note: In the syntax, `estimator` refers to the specific estimator or transformer object from Scikit-Learn that is being used. `X` represents the input data.

Example: Suppose we have a dataset `train_data` for training and `test_data` for testing. We can use `fit()` to learn the parameters from the training data (`estimator.fit(train_data)`) and then use `transform()` to apply the learned transformation to the test data (`transformed_data = estimator.transform(test_data)`). Alternatively, we can use `fit_transform()` to perform both steps in one (`transformed_data = estimator.fit_transform(data)`).

Fine Tuning Linear Regression model using SGDRegressor

The best hyperparameter values for an `SGDRegressor` model depend on your specific dataset and problem. Hyperparameter tuning is typically done using techniques like grid search or random search in combination with cross-validation to find the optimal hyperparameters for your specific task. However, some guidance on commonly tuned hyperparameters for the `SGDRegressor` model:

1. **Learning Rate (`eta0` or `learning_rate`):** This parameter controls the step size during training. Values typically range from 0.01 to 0.1 or smaller. You may need to experiment to find the best learning rate for your specific dataset.

2. **Number of Epochs (`max_iter`):** This parameter determines the number of iterations the algorithm will run for. You should set it to a sufficiently large value to ensure convergence but not too large to avoid overfitting.

3. **Regularization (`alpha`):** The `alpha` parameter controls the amount of regularization applied to the model. Values are typically set logarithmically, e.g., [0.0001, 0.001, 0.01, 0.1, 1.0]. A smaller `alpha` encourages the model to fit the training data closely, while a larger `alpha` encourages a simpler model.

4. **Penalty (`penalty`):** This parameter specifies the type of regularization. Common choices are 'l2' (ridge) and 'l1' (lasso). You can also use 'elasticnet' to combine both 'l2' and 'l1' penalties.

5. **Loss Function (`loss`):** The `loss` parameter determines the loss function used during training. Common options include 'squared_loss' for ordinary least squares (OLS) regression and 'huber' for robust regression.

6. **Shuffling (`shuffle`):** Setting this to `True` can help with convergence by shuffling the training data at the start of each epoch.

7. **Mini-Batch Size (`batch_size`):** You can specify a mini-batch size if you want to use mini-batch gradient descent. A common value is 32 or 64, but this can depend on your dataset size.

8. **Random Seed (`random_state`):** Set a random seed for reproducibility.

Remember that the best hyperparameters depend on the nature of your dataset, its size, and the specific problem you're trying to solve. It's a good practice to perform hyperparameter tuning using techniques like cross-validation to find the best combination of hyperparameters for your particular case. Additionally, you may want to consider feature engineering and data preprocessing as these can also significantly impact your model's performance.

Cross Validation

Cross-validation is a widely used technique in machine learning for assessing the performance and generalization of a predictive model. It helps to estimate how well a model will perform on unseen data by dividing the dataset into multiple subsets, training the model on different subsets, and evaluating its performance on the remaining data. This process helps in detecting issues like overfitting or underfitting and provides a more reliable estimate of a model's performance compared to a single train-test split.

Here are the key steps involved in cross-validation:

1. **Data Splitting:** The dataset is divided into "folds" or "subsets." Common choices include k-fold cross-validation and stratified k-fold cross-validation. In k-fold cross-validation, the data is divided into k equal-sized folds. In stratified k-fold, the division is done while preserving the class distribution, which is useful for classification tasks.

2. **Model Training and Evaluation**: The model is trained and evaluated k times, with each of the k subsets used as the test set exactly once while the remaining k-1 subsets are used as the training set. This results in k sets of evaluation metrics.

3. **Performance Metrics**: Common performance metrics, such as mean squared error (MSE) for regression tasks or accuracy, precision, recall, and F1-score for classification tasks, are computed for each fold.

4. **Performance Aggregation**: The performance metrics from each fold are aggregated to compute a single performance metric that represents the model's overall performance. For example, in k-fold cross-validation, you might calculate the mean or median of the k performance metrics to obtain a more robust estimate.

5. **Hyperparameter Tuning**: Cross-validation is often used for hyperparameter tuning, where different combinations of hyperparameters are evaluated across multiple folds to find the best set of hyperparameters.

Cross Validation

Benefits of Cross-Validation:

- Provides a more robust estimate of a model's performance by using multiple splits of the data.
- Helps detect issues like overfitting or underfitting that may not be apparent with a single train-test split.
- Utilizes the entire dataset for both training and testing, maximizing data utilization.

Common Variations of Cross-Validation:

- **K-Fold Cross-Validation**: The dataset is divided into k equally sized subsets, and each fold is used as the test set exactly once.
- **Stratified K-Fold Cross-Validation**: Like k-fold, but it preserves the class distribution in each fold, ensuring that each class is represented proportionally.
- **Leave-One-Out Cross-Validation (LOOCV)**: Each data point serves as a separate test set, while the remaining data points are used for training. LOOCV is useful for very small datasets but can be computationally expensive.
- **Time Series Cross-Validation**: Designed for time series data, it ensures that training data comes before test data to mimic real-world scenarios where future data is unknown.

Cross-validation is a crucial tool for model assessment and selection, and it helps ensure that your machine learning models generalize well to unseen data.

Classification

Classification: a Machine Learning technique to identify the category of new observations based on training data.

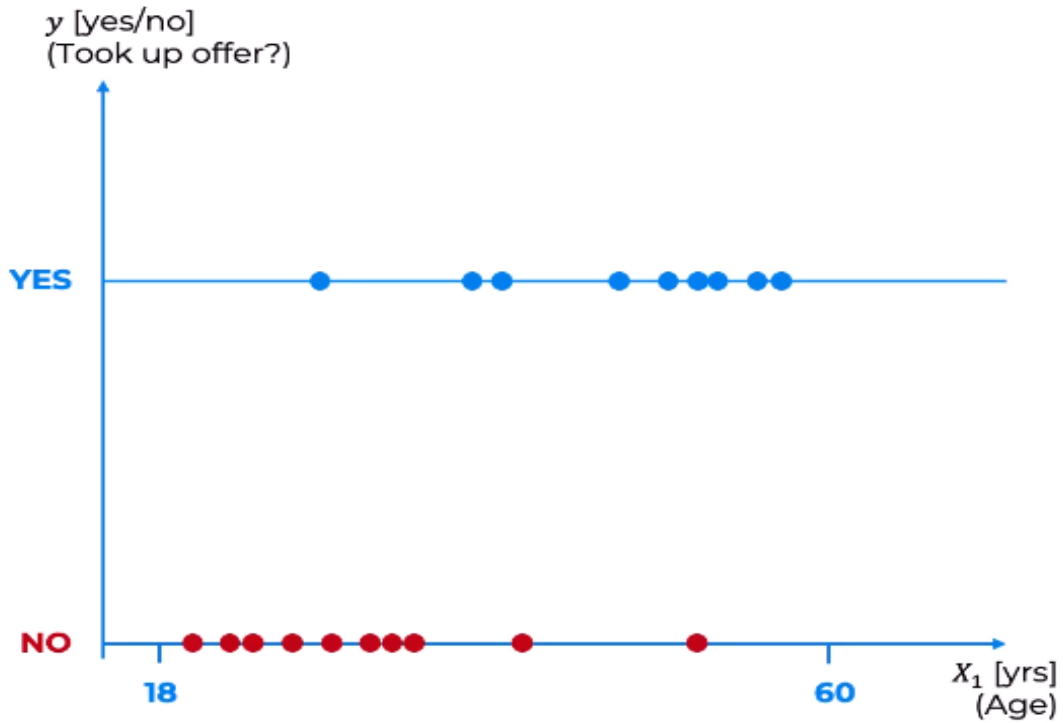
Logistic Regression

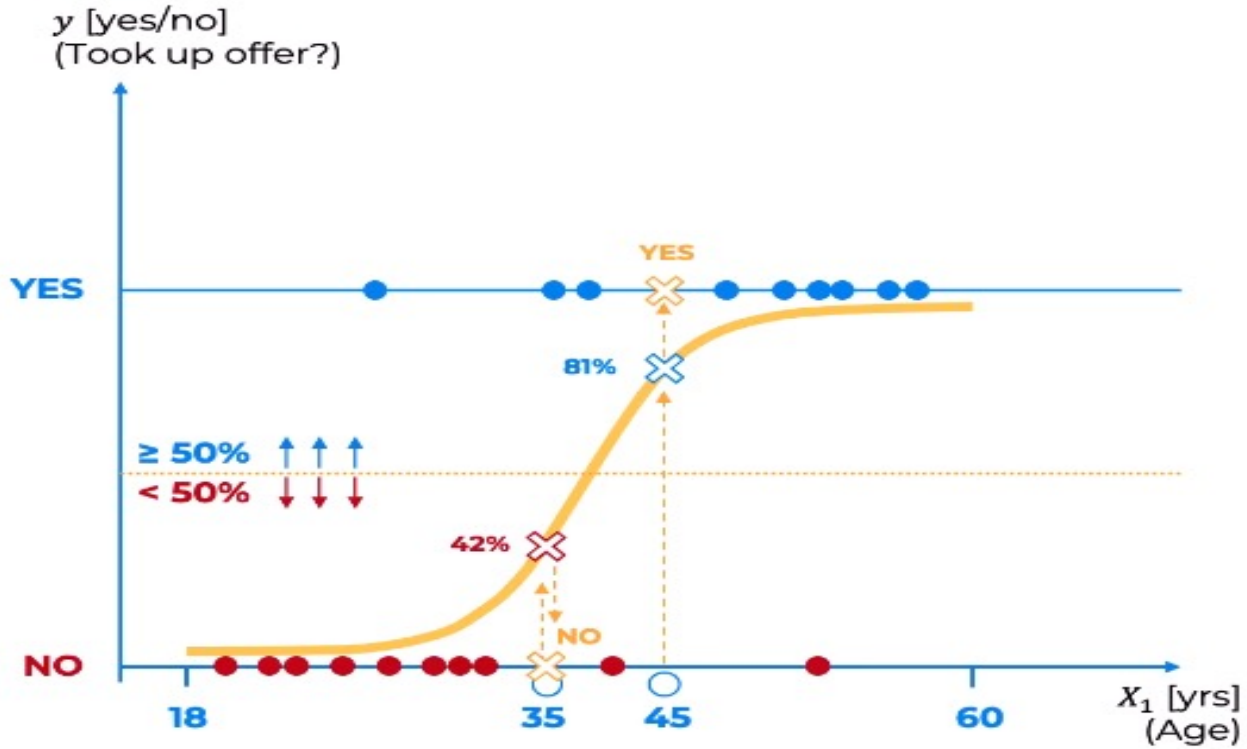
Logistic regression: predict a categorical dependent variable from a number of independent variables.

Will purchase health insurance: Yes / No ~ Age + Income + Level of Education + Family or Single

Will purchase health insurance: Yes / No ~ Age

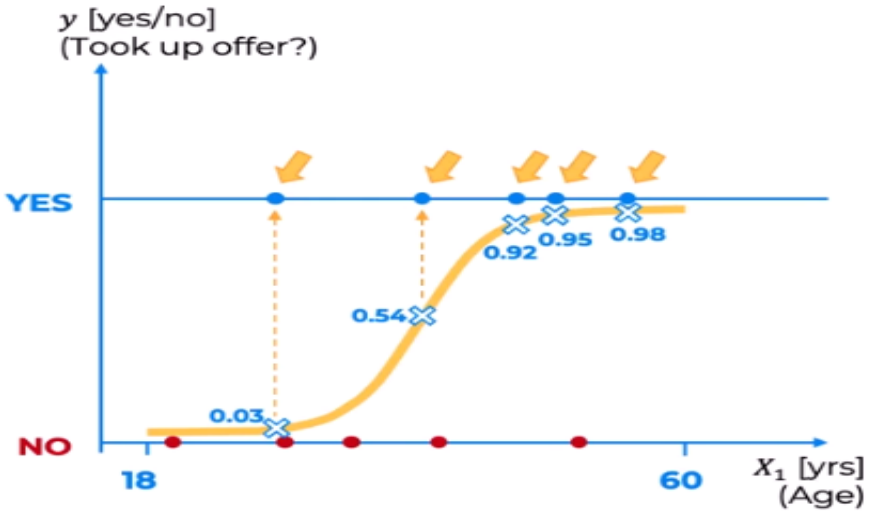
$$\ln \frac{p}{1-p} = b_0 + b_1 X_1$$

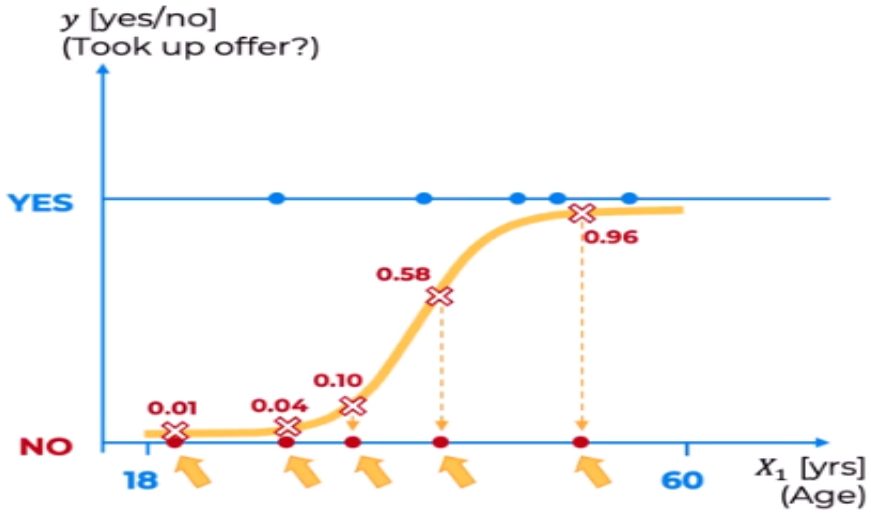




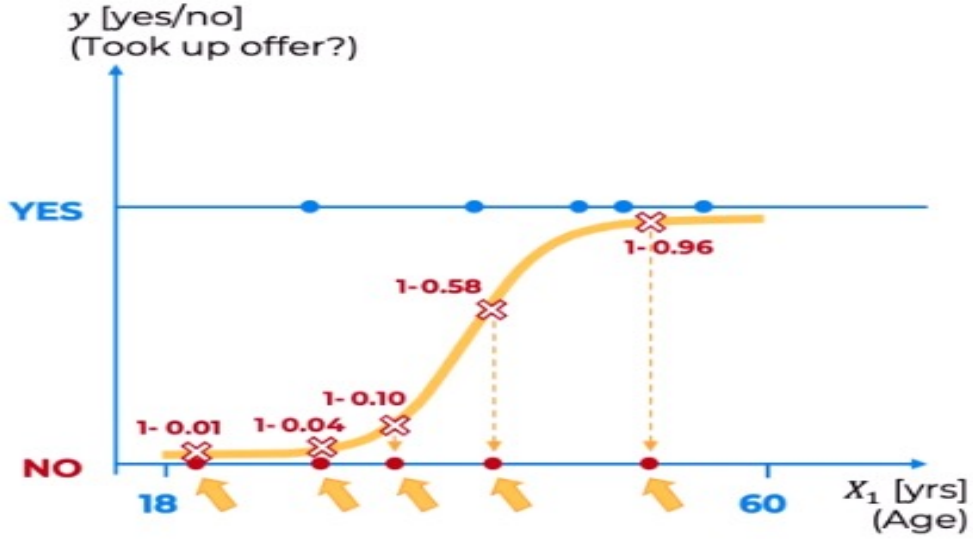
Maximum Likelihood

Probability of saying yes

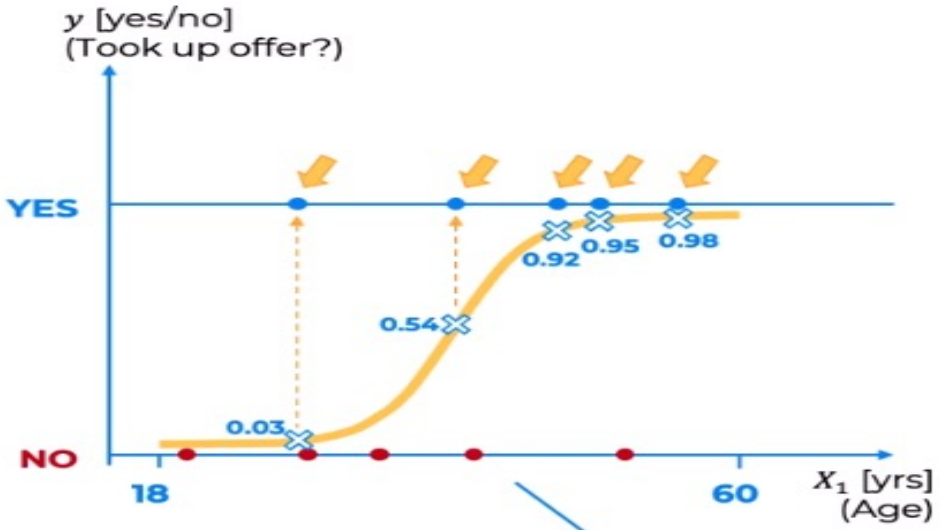


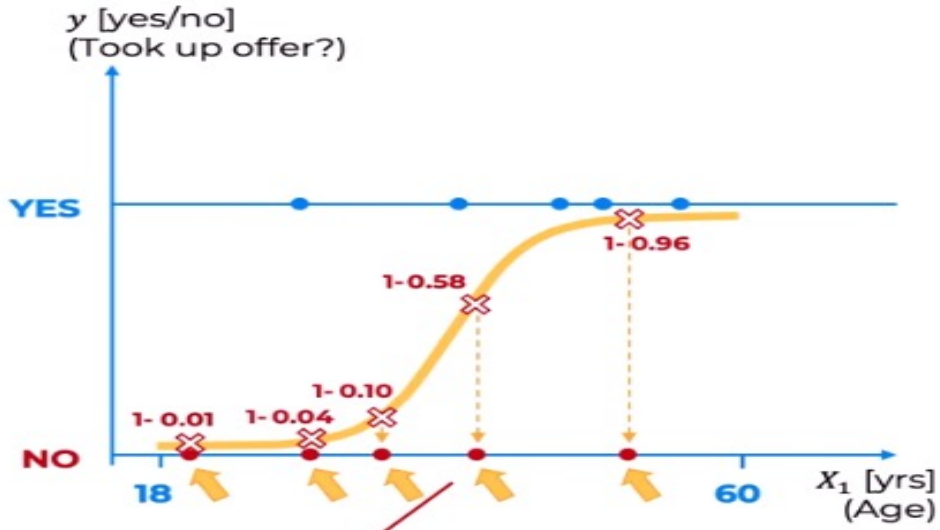


Probability of saying No



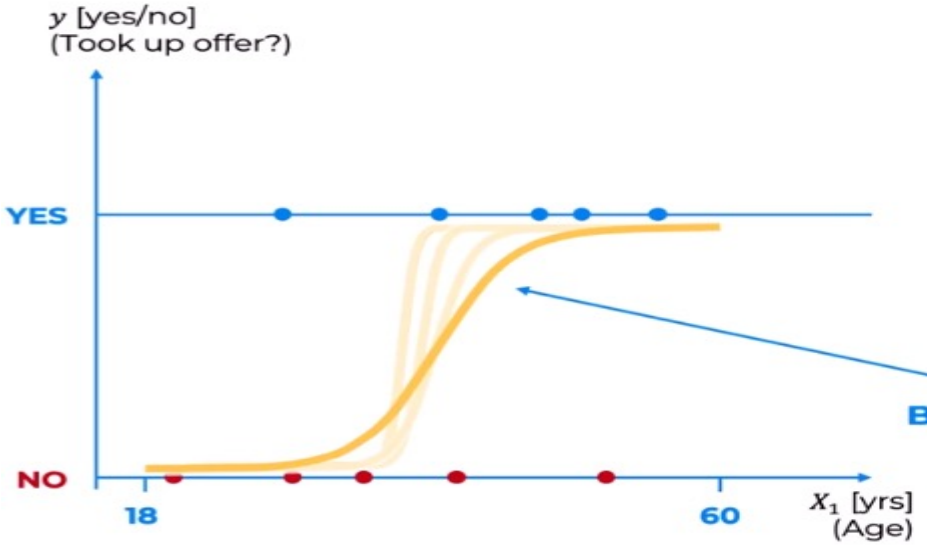
Maximum Likelihood





Likelihood = 0.03 x 0.54 x 0.92 x 0.95 x 0.98 x (1 – 0.01) x (1 – 0.04) x (1 – 0.10) x (1 – 0.58) x (1 – 0.96)

Likelihood = **0.00019939**



Likelihood = 0.00007418

Likelihood = 0.00012845

Likelihood = 0.00016553

Likelihood = 0.00019939

Best Curve <= Maximum Likelihood

Odds Ratio

The odds ratio (OR) is a statistical measure used to quantify the strength and direction of the association between two binary outcomes. It is commonly used in epidemiology, medical research, and social sciences to assess the likelihood of an event occurring in one group compared to another.

The formula for calculating the odds ratio is:

$$OR = \frac{(\text{Odds of the event in group 1})}{(\text{Odds of the event in group 2})}$$

In this formula:

- Group 1 typically represents the group with the outcome of interest or exposure to a certain factor.

- Group 2 represents the reference or comparison group.

The odds of an event occurring in each group are calculated as:

- Odds of the event in group 1 = $\frac{\text{Number of events in group 1}}{\text{Number of non-events in group 1}}$
- Odds of the event in group 2 = $\frac{\text{Number of events in group 2}}{\text{Number of non-events in group 2}}$

The odds ratio can take on values greater than 1, less than 1, or equal to 1, and its interpretation depends on the value:

1. OR = 1: The odds of the event are the same in both groups, indicating no association or difference.

2. OR > 1: Group 1 has higher odds of the event than Group 2, suggesting a positive association or increased risk.

3. OR < 1: Group 2 has higher odds of the event than Group 1, indicating a negative association or decreased risk.

The odds ratio is used to assess the strength and direction of the relationship between two variables and is especially useful when studying the effects of exposures, interventions, or risk factors on outcomes. It is commonly reported in research studies along with confidence intervals to provide a range of values within which the true odds ratio is likely to fall.

Sigmoid Function

ƒ(x) = p(y=1) = e^x / (e^x + 1), is the formula for the logistic function, also known as the sigmoid function.

This function is commonly used in logistic regression and machine learning for binary classification problems.

Here's a breakdown of the components:

- ƒ(x): This represents the logistic function, and it is a function of a real-valued input variable x.
- p(y=1): This represents the probability that a binary outcome variable y equals 1.
- e: This is the base of the natural logarithm, approximately equal to 2.71828.

The logistic function takes a real-valued input x and transforms it into a probability value between 0 and 1. It does so by applying the exponential function (e^x) to x, which makes the output positive, and then dividing it by the sum of that exponential value and 1. This ensures that the output is a valid probability, as probabilities must be between 0 and 1.

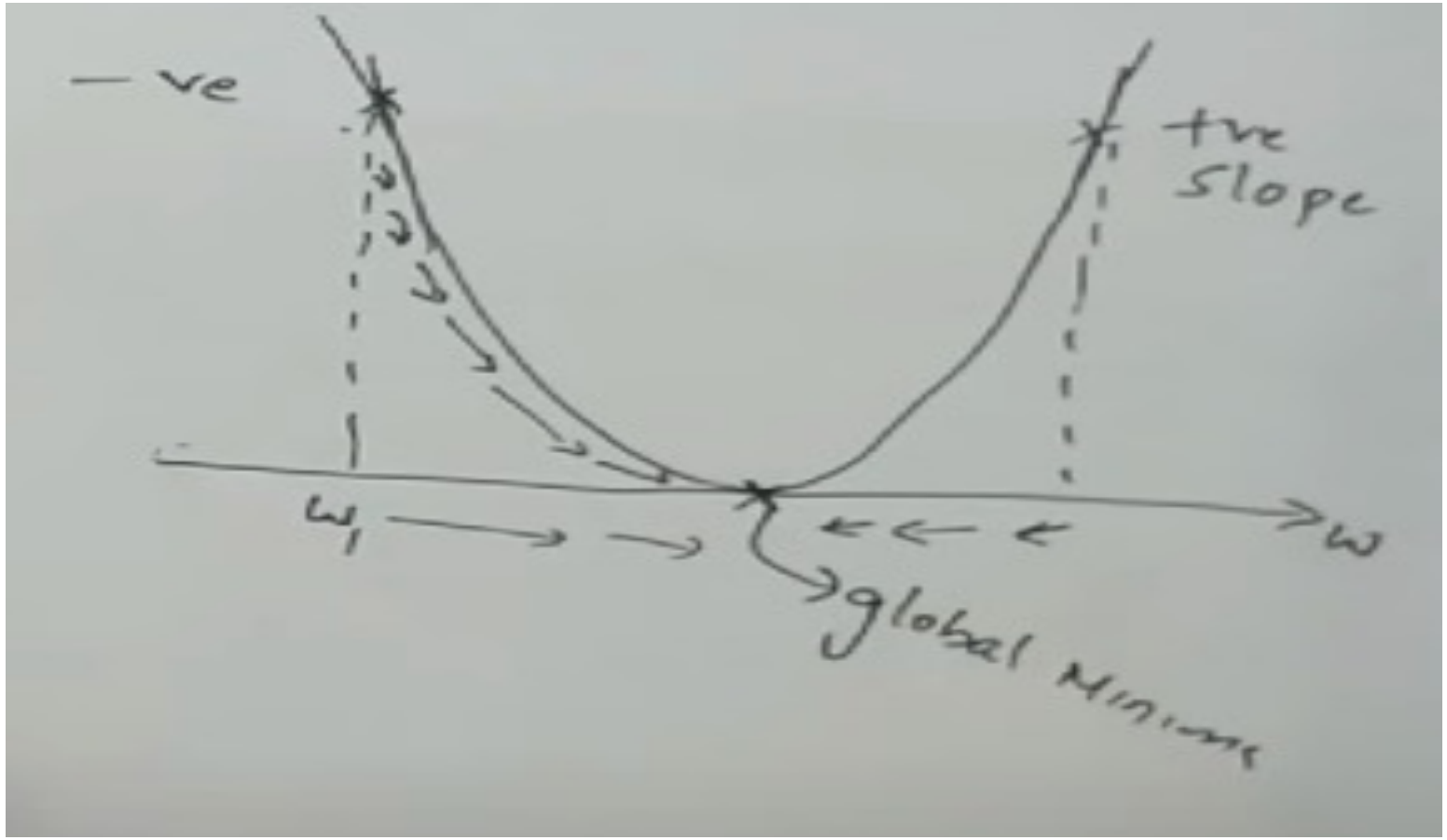
When x is positive, ƒ(x) approaches 1, indicating a high probability of y=1. When x is negative, ƒ(x) approaches 0, indicating a low probability of y=1. When x is zero, ƒ(x) equals 0.5, indicating that y=1 and y=0 are equally likely.

The logistic function is the core component of logistic regression models, where it models the relationship between the input variables and the probability of a binary outcome. It's a useful tool for problems where you want to estimate the probability of an event happening based on one or more predictor variables.

Gradient Descent

Weight Updation Formula- New weight is equal to old weight minus learning rate multiple by derivative(slope) of loss divided by derivative of old weight

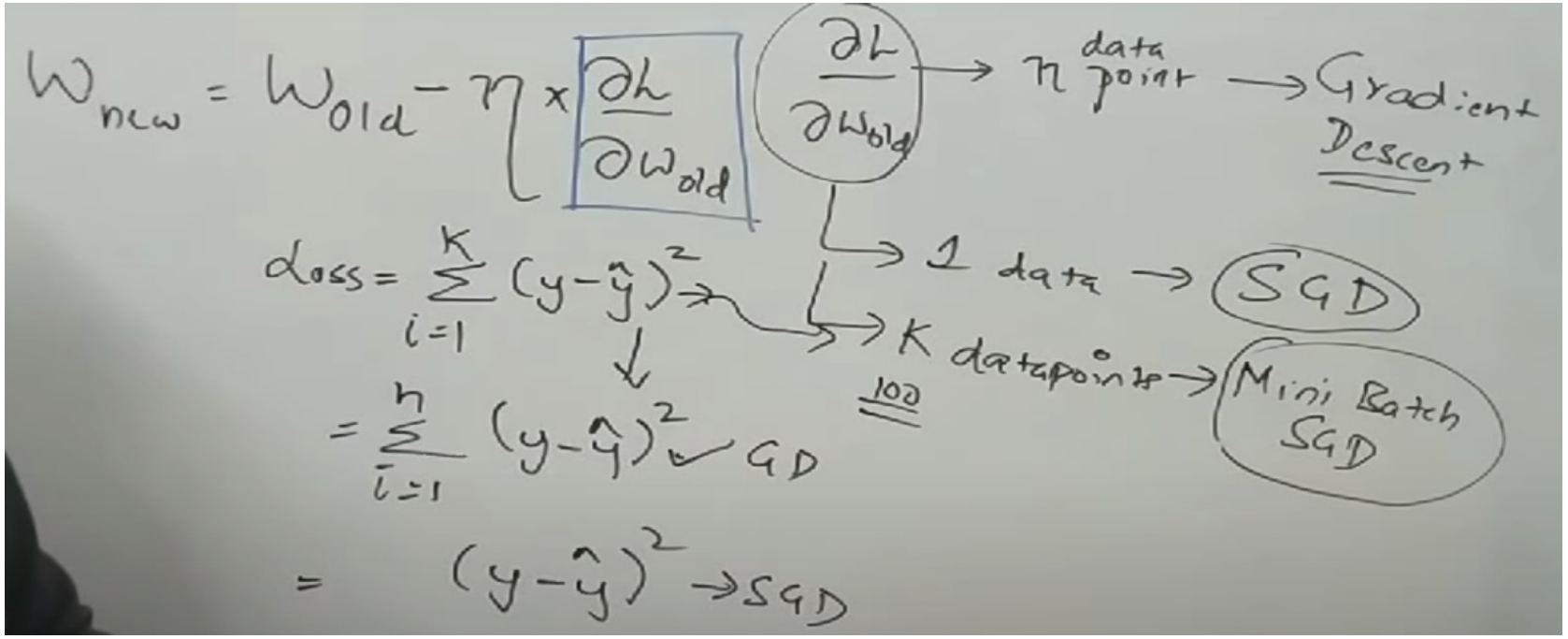
$$W_{new} = W_{old} - \eta \times \frac{\partial h}{\partial W_{old}}$$

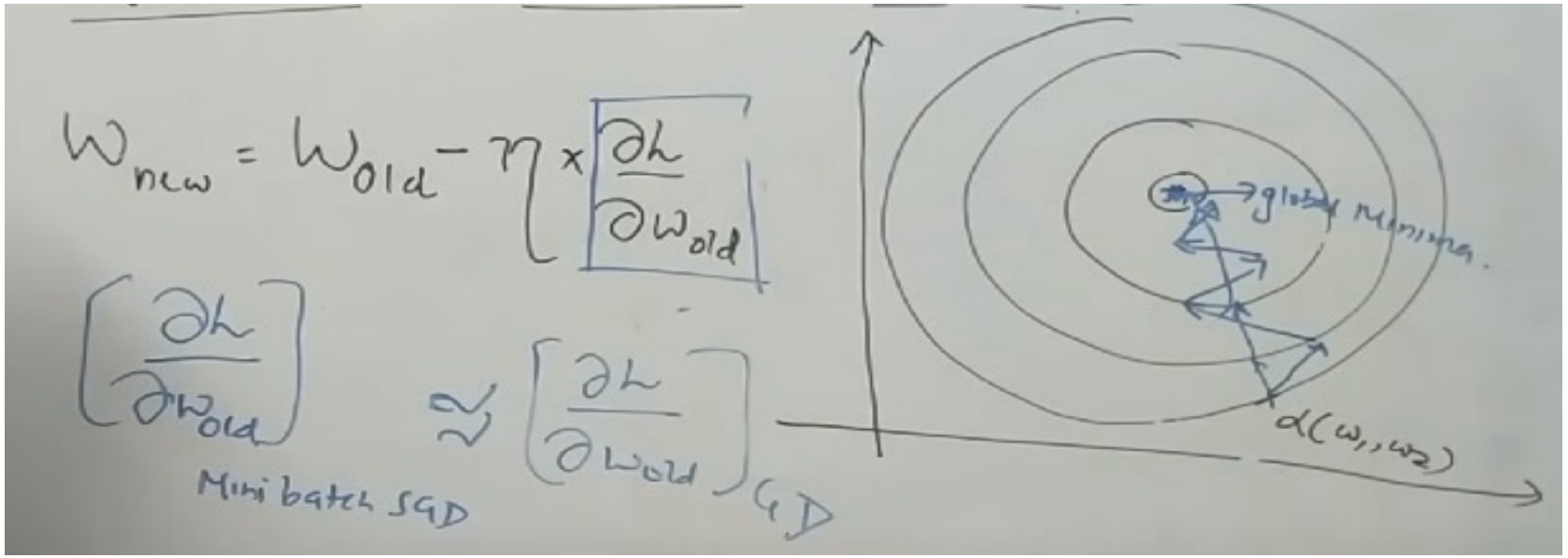


- In traditional gradient descent, the goal is to minimize a cost or loss function J(θ), where θ represents the model parameters (weights and biases).

- Gradient descent iteratively updates θ by taking steps in the direction of the negative gradient of the cost function with respect to θ. This means moving in the direction that reduces the cost.

Gradient Descent and Stochastic Gradient Descent





Gradient Descent and Stochastic Gradient Descent

Advantages:

- SGD is computationally less expensive than batch gradient descent since it processes one training example at a time.
- It can handle large datasets because it doesn't require storing the entire dataset in memory.
- The stochastic nature of the updates can help escape local minima and explore the parameter space more effectively.

Challenges:

- The stochastic updates introduce randomness and can lead to noisy convergence, making the optimization process less stable.
- The learning rate (α) needs to be carefully tuned. A fixed learning rate or a learning rate schedule is often used.
- Since it processes data in random order, convergence may be slower than batch gradient descent.

Confusion Matrix

		Predicted	
		Negative **(N)** **-**	Positive **(P)** **+**
Actual	Negative **-**	True Negative **(TN)**	**False Positive (FP)** **Type I Error**
	Positive **+**	**False Negative (FN)** **Type II Error**	True Positive **(TP)**

Key Metrics

Accuracy, precision, recall, and AUC-ROC are common evaluation metrics used in binary classification tasks to assess the performance of machine learning models. Each metric provides a different perspective on the model's performance, and they are often used together to gain a comprehensive understanding of how well a model is doing.

1. **Accuracy**:
   - Accuracy is a straightforward metric that measures the proportion of correctly classified instances (both true positives and true negatives) out of the total number of instances.
   - Formula: $\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$
   - Accuracy provides an overall assessment of a model's correctness but may not be suitable for imbalanced datasets, where one class dominates.
2. **Precision**:
   - Precision (also known as positive predictive value) measures the proportion of true positive predictions among all positive predictions made by the model.
   - Formula: $\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$
   - Precision is valuable when the cost of false positives is high. It tells us how well the model avoids making incorrect positive predictions.

Key Metrics

3. **Recall**:
   - Recall (also known as sensitivity or true positive rate) measures the proportion of true positive predictions among all actual positive instances.
   - Formula: $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$
   - Recall is valuable when the cost of false negatives is high. It tells us how well the model captures all positive instances.
4. **AUC-ROC (Area Under the Receiver Operating Characteristic Curve)**:
   - AUC-ROC is a metric that evaluates the ability of a binary classification model to distinguish between positive and negative classes across various classification thresholds.
   - It quantifies the area under the ROC curve, where the ROC curve plots the true positive rate (Recall) against the false positive rate at different threshold settings.
   - AUC-ROC ranges from 0 to 1, with higher values indicating better model performance.
   - AUC-ROC is useful for assessing a model's overall discriminatory power and is less affected by class imbalance.

Key Metrics

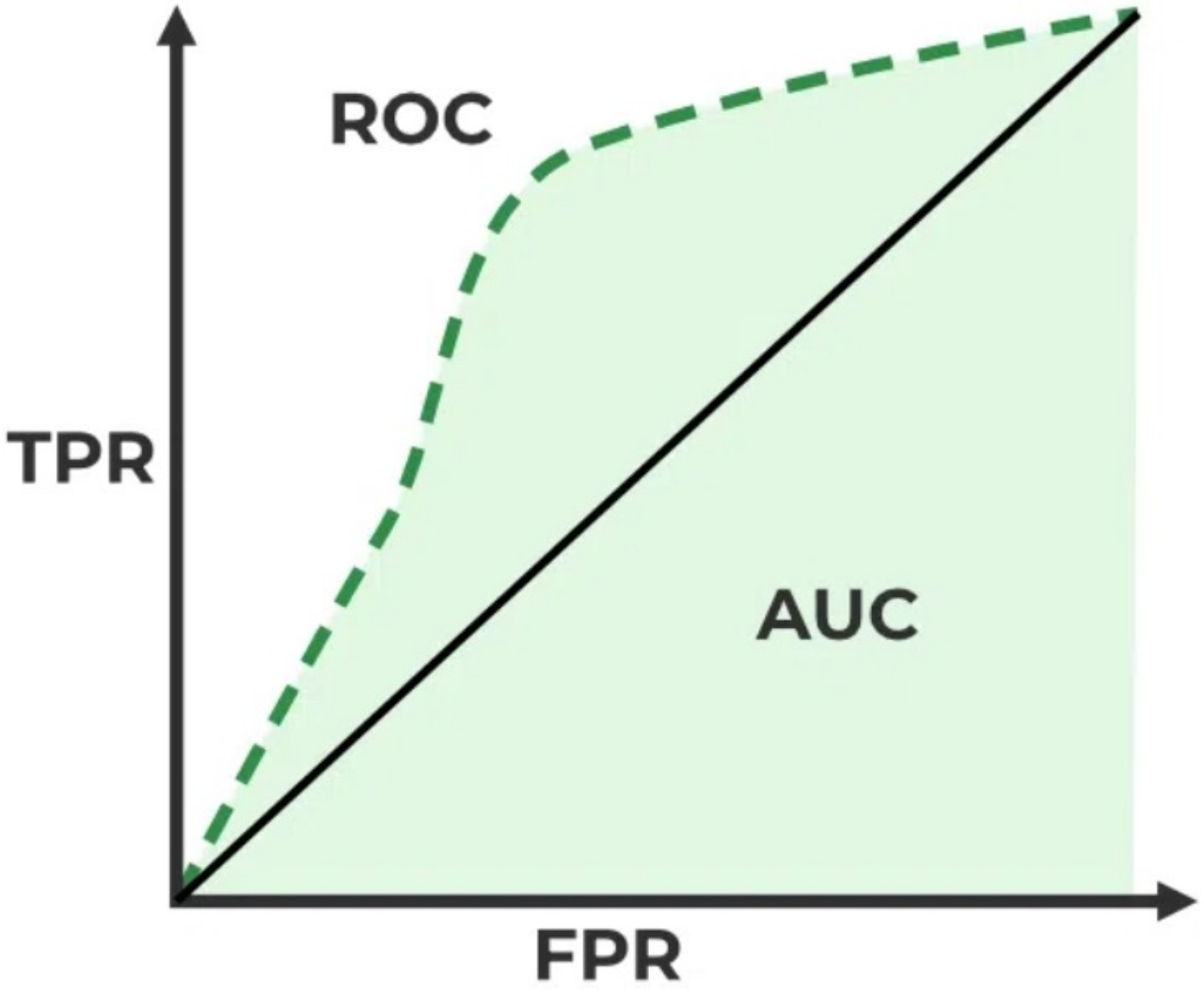
Specificity:

- Specificity, also known as the True Negative Rate or TN Rate, measures the proportion of true negatives (correctly predicted negatives) out of all actual negatives.
- It quantifies how well the model can correctly identify instances of the negative class.
- Formula: $\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$
- Specificity ranges from 0 to 1, with higher values indicating better performance in correctly identifying negatives.

Error Rate:

- The error rate (or misclassification rate) represents the proportion of misclassified instances, i.e., instances that were classified incorrectly.
- Formula: $\text{Error Rate} = \frac{\text{False Positives} + \text{False Negatives}}{\text{Total Number of Instances}}$
- Error rate provides an overall measure of how often the model's predictions are incorrect.

ROC AUC Curve



ROC Curve

ROC stands for Receiver Operating Characteristics, and the ROC curve is the graphical representation of the effectiveness of the binary classification model. It plots the true positive rate (TPR) vs the false positive rate (FPR) at different classification thresholds.

AUC Curve:

AUC stands for Area Under the Curve, and the AUC curve represents the area under the ROC curve. It measures the overall performance of the binary classification model. As both TPR and FPR range between 0 to 1, So, the area will always lie between 0 and 1, and A greater value of AUC denotes better model performance. Our main goal is to maximize this area in order to have the highest TPR and lowest FPR at the given threshold. The AUC measures the probability that the model will assign a randomly chosen positive instance a higher predicted probability compared to a randomly chosen negative instance. ***AUC measures how well a model is able to distinguish between classes.***

ROC AUC Curve

And as said earlier ROC is nothing but the plot between TPR and FPR across all possible thresholds and AUC is the entire area beneath this ROC curve.