Linear Regression VS Logistic Regression

Linear Regression:

- ➔ Linear regression is used when the target variable (the variable you are trying to predict) is continuous.
- ➔ It models the relationship between one or more independent variables (features) and a continuous dependent variable by fitting a linear equation to observed data.

The equation of a simple linear regression model with one independent variable is:
$y = mx + b$, where (y) is the dependent variable, (x) is the independent variable, (m) is the slope of the line, and (b) is the y-intercept.

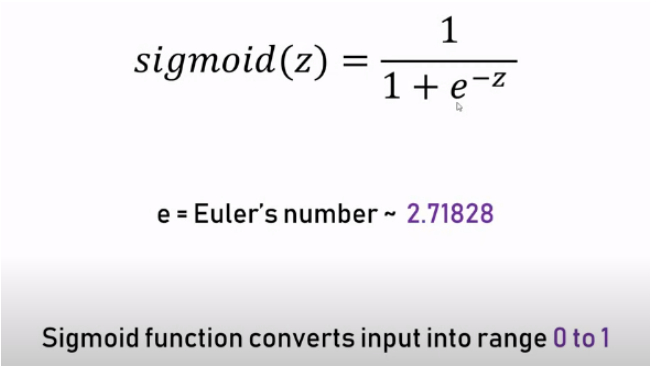
Linear regression is suitable for predicting outcomes where the relationship between the independent and dependent variables is linear.

Logistic Regression:

- Logistic regression is used when the target variable is categorical with two or more levels (binary or multi-class classification).
- It models the probability that the dependent variable belongs to a particular category.
- Unlike linear regression, logistic regression uses the logistic function (also called the sigmoid function) to model the relationship between the independent and dependent variables.
- The logistic function maps any real-valued number into the range of 0 to 1, which is interpreted as the probability of the dependent variable being in one category or another.

The equation of logistic regression is:

$$y = \frac{1}{1 + e^{-(m*x+b)}}$$



- Logistic regression is widely used in various fields such as healthcare (predicting the likelihood of disease), marketing (predicting customer churn), and finance (predicting loan default).

In summary, linear regression is used for continuous outcomes, while logistic regression is used for categorical outcomes.

Over Fitting Under Fitting

When our model works well in train data but not with test data that's called overfitting.
When our model works well in test data but not with train data that's called underfitting.

Standardize or Normalize

StandardScaler is a preprocessing technique used in machine learning to standardize or normalize the features (independent variables) of your dataset

It transforms the data in such a way that it has a mean of 0 and a standard deviation of 1, making it easier for certain machine-learning algorithms to - perform well

Here's how it works:

Mean Centering: StandardScaler first calculates the mean (average) of each feature across the dataset.

Scaling: It then scales each feature by subtracting the mean and dividing by the standard deviation. This process transforms the data into a - distribution with a mean of 0 and a standard deviation of 1.

The formula for standardizing a feature "x" using StandardScaler is:

x_standardized =(x - mean(x)) / std(x)

“x” is an individual data point in the feature

“mean(x)” is the mean of the feature

“std(x)” is the standard deviation of the feature

Normalization: It ensures that all features have similar scales, preventing some features from dominating others in algorithms sensitive to the magnitude of values, like gradient descent in many machine learning models.

Numerical Stability: Scaling can help with numerical stability in some optimization algorithms, making convergence faster and more reliable.

Interpretability: Standardized features can be more interpretable because they are on a common scale, allowing for direct comparisons of the impact of each feature on the model.

StandardScaler is often used as a preprocessing step in machine learning pipelines, along with other data preprocessing and feature engineering techniques, to improve the performance and reliability of various algorithms, such as linear regression, support vector machines, and k-nearest neighbors.

Regularization

We use regularization to reduce. overfitting. when a model learns the training data too well, capturing noise or irrelevant patterns that do not generalize well to unseen data.

There are several types of regularization techniques, we can use to reduce overfitting::

L1 Regularization (Lasso Regression):

- ➔ L1 helps with feature selection and can set some feature coefficients to zero.
- ➔ L1 regularization encourages sparse solutions and can be useful when dealing with high-dimensional data or when there are many irrelevant features.

L2 Regularization (Ridge Regression):

- ➔ This penalizes large coefficients while still keeping them non-zero.
- ➔ L2 regularization tends to spread the impact of the regularization across all features, rather than eliminating some. It can help mitigate multicollinearity and stabilize the model.

Elastic Net Regularization:

Elastic Net regularization combines both L1 and L2 penalties, allowing for a more flexible regularization approach. It includes the L1 and L2 penalties in the loss function, controlled by two hyperparameters.

Dropout Regularization:

- Dropout is a technique commonly used in neural networks. During training, randomly selected neurons are temporarily removed ("dropped out") with a certain probability. This forces the network to learn redundant representations and prevents overreliance on individual neurons.
- Dropout regularization helps prevent overfitting by promoting robustness and generalization in neural network models.

Hyperparameter Tuning

Where we tune parameters for our best model performance.
GridSearchCV: *Each combination will try.*
RandomizedSearchCV: *Some random combination will try and we use it for huge data processing.*