PW SKILLS

Lesson Plan

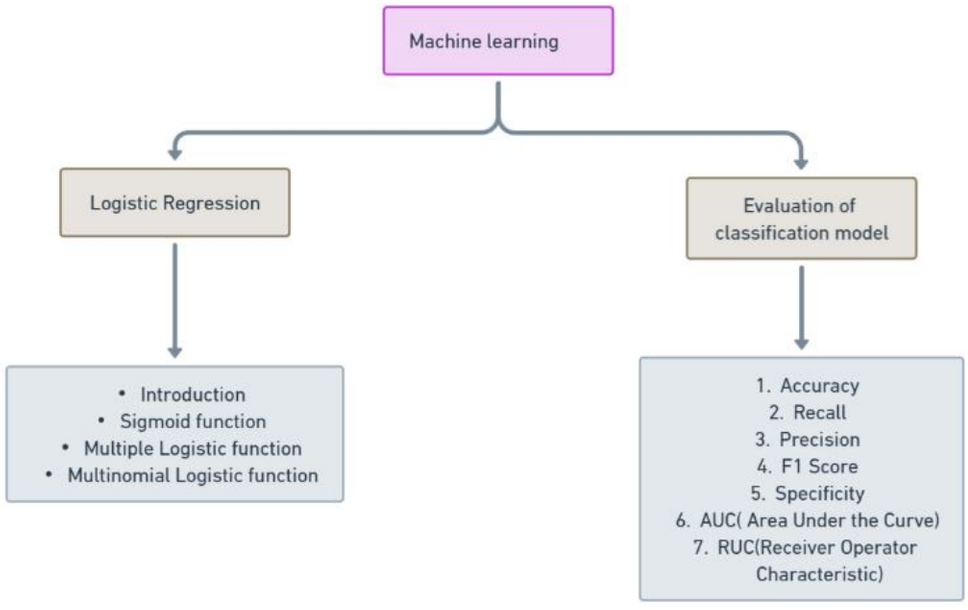
Logistic Regression

Topics Covered

- Logistic Regression
- Sigmoid function
- Multiple Logistic function
- Multinomial Logistic function
- Evaluation of classification model

1. Accuracy
2. Recall
3. Precision
4. F1 Score
5. Specificity
6. AUC(Area Under the Curve)
7. RUC(Receiver Operator Characteristic)

- Hyperparameter tuning
- Advantages and disadvantages of LR



Introduction

In linear regression, the type of data we deal with is quantitative, whereas we use classification models to deal with qualitative data or categorical data. The algorithms used for solving a classification problem first predict the probability of each of the categories of the qualitative variables, as the basis for making the classification. And, as the chances are continuous numbers, classification using probabilities also behaves like regression methods. Logistic regression is one such type of classification model that is used to classify the dependent variable into two or more classes or categories.

Why don't we use Linear regression for classification problems?

Let's suppose you took a survey and noted the response of each person as satisfied, neutral, or Not satisfied. Let's map each category:

Satisfied – 2

Neutral – 1

Not Satisfied – 0

But this doesn't mean that the gap between Not satisfied and Neutral is the same as between Neutral and satisfied. There is no mathematical significance in this mapping. We can also map the categories like:

Satisfied – 0

Neutral – 1

Not Satisfied – 2

It's completely fine to choose the above mapping. If we apply linear regression to both types of mappings, we will get different sets of predictions. Also, we can get prediction values like 1.2, 0.8, 2.3, etc. which makes no sense for categorical values. So, there is no normal method to convert qualitative data into quantitative data for use in linear regression. Although, for binary classification, i.e. when there are only two categorical values, using the least square method can give decent results. Suppose we have two categories Black and White and we map them as follows:

Black – 0

White - 1

We can assign predicted values for both the categories such as Y> 0.5 going to class white and vice versa. Although, there will be some predictions for which the matter can be greater than 1 or less than 0 making them hard to classify in any class. Nevertheless, linear regression can work decently for binary classification but not well for multi-class sorting. Hence, we use classification methods for dealing with such problems.

Logistic Regression

Logistic regression is one such regression algorithm that can be used for performing classification problems. It calculates the probability that a given value belongs to a specific class. If the probability is more than 50%, it assigns the value to that particular class else if the probability is less than 50%, the value is assigned to the other class. Therefore, logistic regression acts as a binary classifier.
Working on a Logistic Model

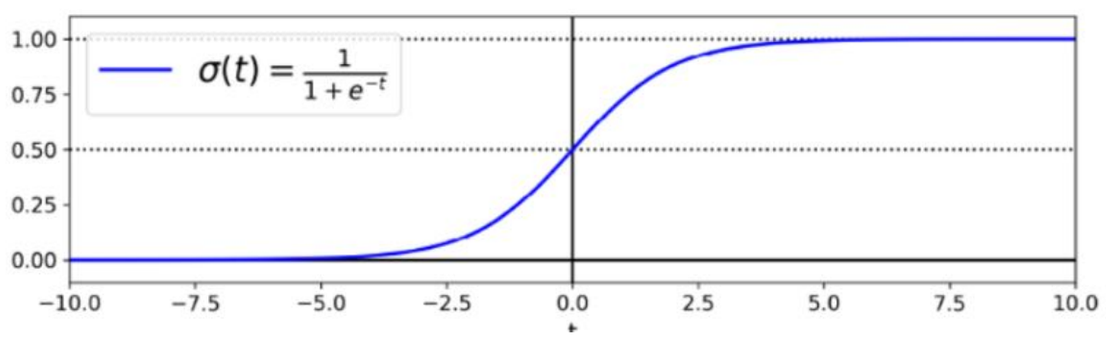
For linear regression, the model is defined by:

$$y = \beta_0 + \beta_1 x \text{ - (i)}$$

and for logistic regression, we calculate probability, i.e. y is the probability of a given variable x belonging to a certain class. Thus, it is obvious that the value of y should lie between 0 and 1.
But, when we use equation(i) to calculate probability, we would get values less than 0 as well as greater than 1. That doesn't make any sense. So, we need to use an equation that always gives values between 0 and 1, as we desire while calculating the probability.

Sigmoid function

We use the sigmoid function as the underlying function in Logistic regression. Mathematically and graphically, it is shown as:



Why do we use the Sigmoid Function?

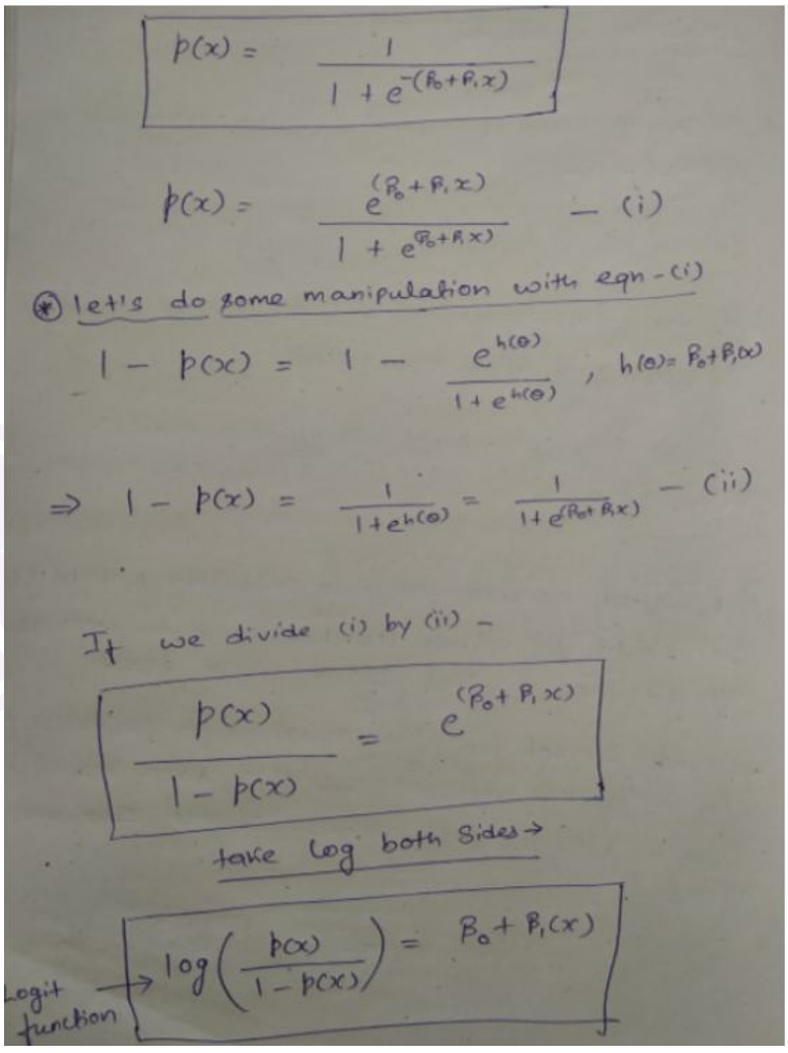
1) The sigmoid function's range is bounded between 0 and 1. Thus it's useful in calculating the probability of the Logistic function. 2) Its derivative is easy to calculate than other functions which is useful during gradient descent calculation. 3) It is a simple way of introducing non-linearity to the model.

Although there are other functions as well, which can be used, sigmoid is the most common function used for logistic regression. We will talk about the rest of the functions in the neural network section.

The logistic function is given as:

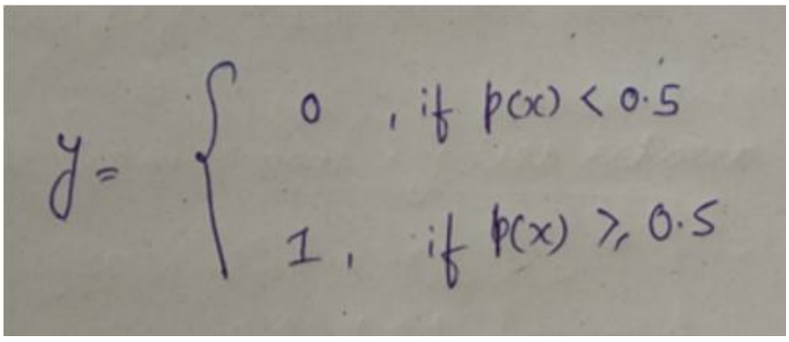
$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

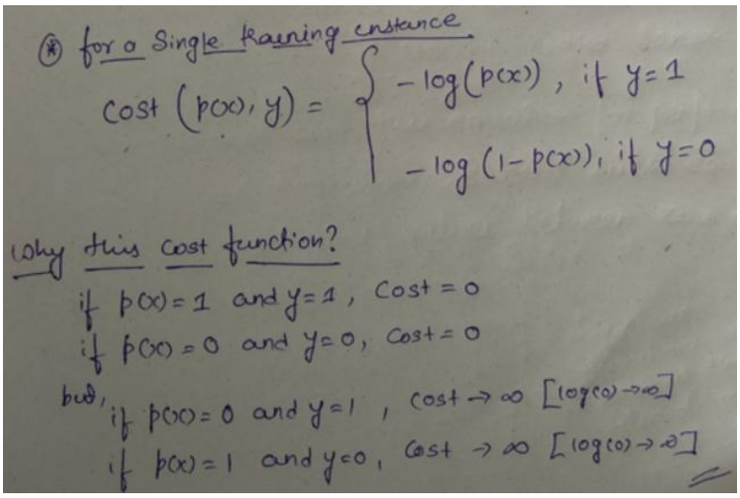
Let's see some manipulation with the logistic function:



We can see that the logit function is linear in terms of x.

Prediction





The cost function for the whole training set is given as :

$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1-y^{(i)}\right)log\left(1-\hat{p}^{(i)}\right)\right]$$

The values of parameters (θ) for which the cost function is minimum are calculated using the gradient descent (as discussed in the Linear Regression section) algorithm. The partial derivative for the cost function is given as :

$$\frac{\partial}{\partial\theta_j}J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m}\left(\sigma\left(\boldsymbol{\theta}^T\mathbf{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

Multiple Logistic Function

We can generalize the simple logistic function for multiple features as:

$$p(X) = \frac{e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}}{1+e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}}.$$

And the logit function can be written as:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0+\beta_1X_1+\cdots+\beta_pX_p,$$

The coefficients are calculated the same we did for the simple logistic functions, by passing the above equation in the cost function.

Just like we did in multilinear regression, we will check for correlation between different features for Multi logistics as well.

We will see how we implement all the above concepts through a practical example

Multinomial Logistics Regression(Number of Labels >2)

Many times, there are classification problems where the number of classes is greater than 2. We can extend Logistic regression for multi-class classification. The logic is simple; we train our logistic model for each class and calculate the probability(hθx) that a specific feature belongs to that class. Once we have trained the model for all the classes, we predict a new value's class by choosing that class for which the probability(hθx) is maximum. Although we have libraries that we can use to perform multinomial logistic regression, we rarely use logistic regression for classification problems where the number of classes is more than 2. There are many other classification models for such scenarios. We will see more of that in the coming lectures.

Learning Algorithm

The learning algorithm is how we search the set of possible hypotheses (hypothesis space H) for the best parameterization (in this case the weight vector w). This search is an optimization problem looking for the hypothesis that optimizes an error measure.
There is no sophisticated, closed-form solution like least-squares linear, so we will use gradient descent instead. Specifically, we will use batch gradient descent which calculates the gradient from all data points in the data set.

Luckily, our "cross-entropy" error measure is convex so there is only one minimum. Thus the minimum we arrive at is the global minimum.

To learn we're going to minimize the following error measure using batch gradient descent.

$$e(h(\mathbf{x}_n), y_n) = \ln\left(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}\right)$$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} e(h(\mathbf{x}_n), y_n) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}\right)$$

We'll need the derivative of the point loss function and possibly some abuse of notation.

$$\frac{d}{d\mathbf{w}} e(h(\mathbf{x}_n), y_n) = \frac{-y_n \mathbf{x}_n e^{-y_n \mathbf{w}^T \mathbf{x}_n}}{1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}} = -\frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}}$$

With the point loss derivative, we can determine the gradient of the in-sample error:

$$\begin{aligned}
\nabla E_{\text{in}}(\mathbf{w}) &= \frac{d}{d\mathbf{w}}\left[\frac{1}{N}\sum_{n=1}^{N} e(h(\mathbf{x}_n), y_n)\right] \\
&= \frac{1}{N}\sum_{n=1}^{N} \frac{d}{d\mathbf{w}} e(h(\mathbf{x}_n), y_n) \\
&= \frac{1}{N}\sum_{n=1}^{N} \left(-\frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}}\right) \\
&= -\frac{1}{N}\sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}}
\end{aligned}$$

Our weight update rule per batch gradient descent becomes
where η is the learning rate.

The Objective of the Model

The objective of a Machine Learning (ML) model is to make accurate predictions or decisions based on input data. It aims to learn patterns and relationships from historical data and generalize that knowledge to make predictions on new, unseen data. The primary goal is to minimize the model's prediction error and maximize its performance on the task it is designed for, such as classification, regression, clustering, or reinforcement learning. Ultimately, the objective is to leverage the power of data and algorithms to solve real-world problems, automate tasks, gain insights, and optimize decision-making processes.

Evaluation of a Classification Model

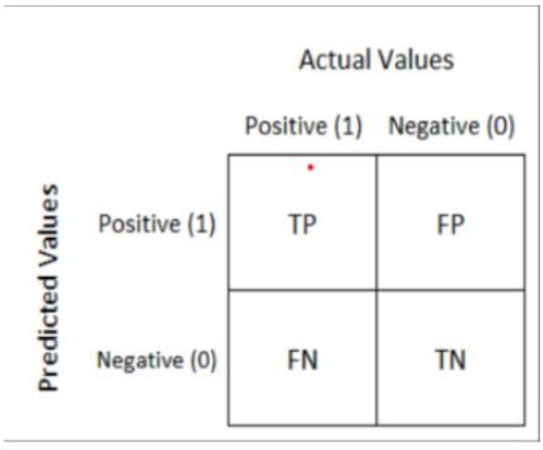
In machine learning, once we have a result of the classification problem, how do we measure how accurate our classification is? For a regression problem, we have different metrics like R Squared score, Mean Squared Error, etc. What are the metrics to measure the credibility of a classification model?
Metrics In a regression problem, the accuracy is generally measured in terms of the difference between the actual values and the predicted values. In a classification problem, the credibility of the model is measured using the confusion matrix generated, i.e., how accurately the true positives and true negatives were predicted.

The different metrics used for this purpose are:

- Accuracy
- Recall
- Precision
- F1 Score
- Specificity
- AUC(Area Under the Curve)
- RUC(Receiver Operator Characteristic)

Confusion Matrix

A typical confusion matrix looks like the figure shown.



Where the terms have the meaning:

True Positive(TP): A result that was predicted as positive by the classification model and also is positive
True Negative(TN): A result that was predicted as negative by the classification model and also is negative
False Positive(FP): A result that was predicted as positive by the classification model but actually is negative
False Negative(FN): A result that was predicted as negative by the classification model but actually is positive.

The Credibility of the model is based on how many correct predictions did the model do.

Accuracy

The mathematical formula is :

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Or, it can be said that it's defined as the total number of correct classifications divided by the total number of classifications.

Recall or Sensitivity

The mathematical formula is:

$$\text{Recall} = \frac{TP}{(TP+FN)}$$

Or, as the name suggests, it is a measure of the total number of positive results and how many positives were correctly predicted by the model.

It shows how relevant the model is, in terms of positive results only.

Let's suppose in the previous model, the model gave 50 correct predictions(TP) but failed to identify 200 cancer patients(FN). Recall in that case will be:

Recall=50/(50+200) = 0.2 (The model was able to recall only 20% of the cancer patients)

Precision

Precision is a measure of amongst all the positive predictions, how many of them were actually positive. Mathematically,

$$\text{Precision} = \frac{TP}{(TP+FP)}$$

Let's suppose in the previous example, the model identified 50 people as cancer patients(TP) but also raised a false alarm for 100 patients(FP). Hence,

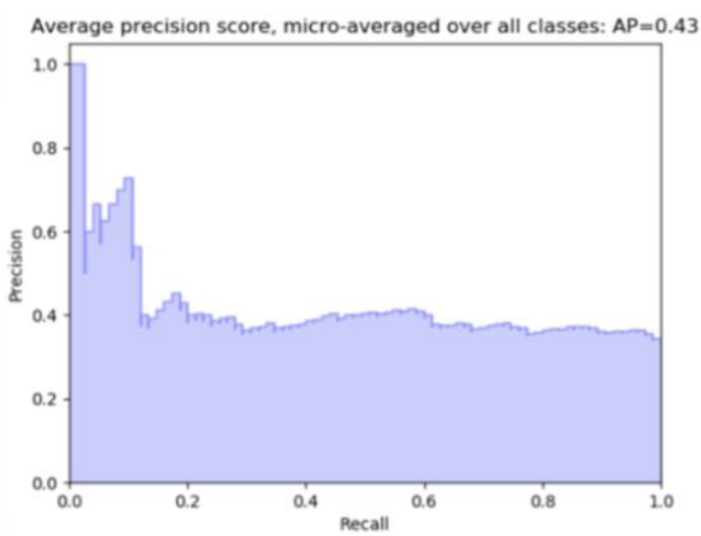
$$\text{Precision} = \frac{50}{(50+100)} = 0.33$$ (The model only has a precision of 33%)

But we have a problem!!

As evident from the previous example, the model had a very high Accuracy but performed poorly in terms of Precision and Recall. So, necessarily Accuracy is not the metric to use for evaluating the model in this case.

Imagine a scenario, where the requirement was that the model recalled all the defaulters who did not pay back the loan. Suppose there were 10 such defaulters and to recall those 10 defaulters, and the model gave you 20 results out of which only the 10 are the actual defaulters. Now, the recall of the model is 100%, but the precision goes down to 50%.

A Trade-off?



As observed from the graph, with an increase in the Recall, there is a drop in the Precision of the model.

So the question is - what to go for? Precision or Recall?

Well, the answer is: it depends on the business requirement.

For example, if you are predicting cancer, you need a 100 % recall. But suppose you are predicting whether a person is innocent or not, you need 100% precision.

Can we maximize both at the same time? No
So, there is a need for a better metric then?
Yes. And it's called an F1 Score

F1 Score

From the previous examples, it is clear that we need a metric that considers both Precision and Recall for evaluating a model. One such metric is the F1 score.

F1 score is defined as the harmonic mean of Precision and Recall.

The mathematical formula is:

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

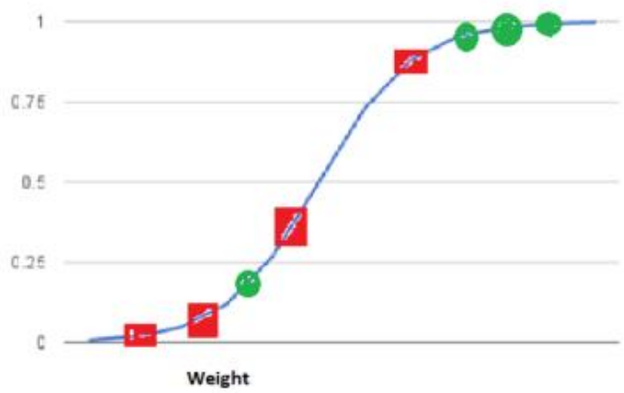
Specificity or True Negative Rate

This represents how specific is the model while predicting the True Negatives. Mathematically,

$$\text{Specificity} = \frac{TN}{(TN+FP)}$$

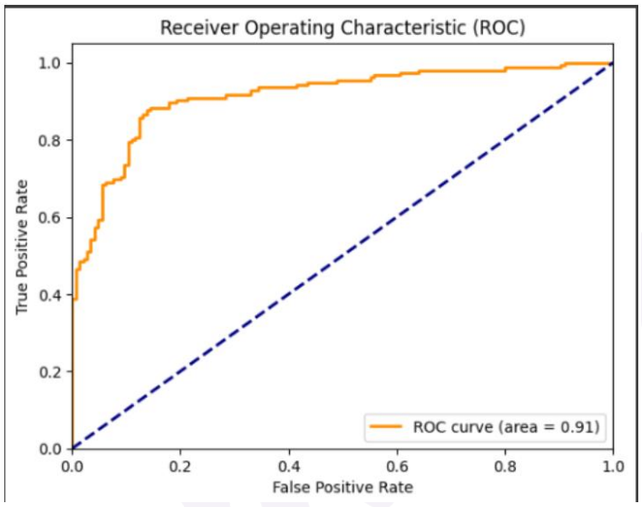
Or, it can be said that it quantifies the total number of negatives predicted by the model with respect to the total number of actual negative or unfavorable outcomes.
Similarly, the False Positive rate can be defined as: (1- specificity) Or, $\frac{FP}{(TN+FP)}$

The following diagram shows a typical logistic regression curve.



- The horizontal lines represent the various values of thresholds ranging from 0 to 1.
- Let's suppose our classification problem was to identify obese people from the given data.
- The green markers represent obese people and the red markers represent to non-obese people.
- Our confusion matrix will depend on the value of the threshold chosen by us.
- For Example, if 0.25 is the threshold then
- TP(actually obese)=3
  TN(Not obese)=2
  FP(Not obese but predicted obese)=2(the two red squares above the 0.25 line)
  FN(Obese but predicted as not obese)=1(Green circle below 0.25line)

A typical ROC curve looks like the following figure.

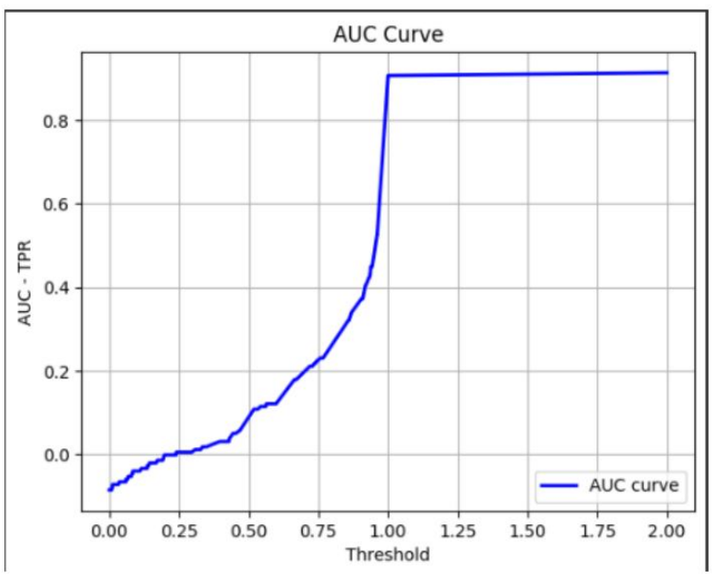


- Mathematically, it represents the various confusion matrices for various thresholds. Each black dot is one confusion matrix.
- The green dotted line represents the scenario when the true positive rate equals the false positive rate.
- As evident from the curve, as we move from the rightmost dot towards the left, after a certain threshold, the false positive rate decreases.
- After some time, the false positive rate becomes zero.
- Mathematically, it represents the various confusion matrices for various thresholds. Each black dot is one confusion matrix.
- The green dotted line represents the scenario when the true positive rate equals the false positive rate.
- As evident from the curve, as we move from the rightmost dot towards the left, after a certain threshold, the false positive rate decreases.
- After some time, the false positive rate becomes zero.

But we have a confusion!!

Let's suppose that we used different classification algorithms, and different ROCs for the corresponding algorithms have been plotted. The question is: which algorithm to choose now? The answer is to calculate the area under each ROC curve.

AUC(Area Under Curve)



- It helps us to choose the best model amongst the models for which we have plotted the ROC curves
- The best model is the one that encompasses the maximum area under it.
- In the adjacent diagram, amongst the two curves, the model that resulted in the red one should be chosen as it clearly covers more area than the blue one

What is the significance of the Roc curve and AUC?

In real life, we create various models using different algorithms that we can use for classification purposes. We use AUC to determine which model is the best one to use for a given dataset. Suppose we have created Logistic regression, SVM as well as a clustering model for classification purposes. We will calculate AUC for all the models separately. The model with the highest AUC value will be the best model to use.

Hyperparameter Tuning in Logistic Regression:

Here's a description of how to test different hyperparameter configurations in Logistic Regression:

Logistic Regression is a binary classification algorithm that models the relationship between input features and a binary target variable using the logistic (sigmoid) function. While Logistic Regression itself does not have many hyperparameters to tune, there is one significant hyperparameter related to regularization:

Penalty:

L1 Regularization (Lasso):
L1 regularization adds the absolute value of the coefficients as a penalty term. It encourages the model to set some coefficients to exactly zero, effectively performing feature selection. This helps in selecting the most important features and making the model more interpretable.

L2 Regularization (Ridge):
L2 regularization adds the square of the coefficients as a penalty term. It discourages large coefficients and leads to more balanced coefficients across all features. L2 regularization helps in reducing the impact of multicollinearity between features and improving the model's stability.

Regularization Strength (C):

Regularization is used to prevent overfitting and improve generalization. In Logistic Regression, two common regularization terms are L1 (Lasso) and L2 (Ridge)

regularization. The strength of regularization is controlled by the hyperparameter "C." Higher values of C indicate weaker regularization (closer to no regularization), and lower values indicate stronger regularization. Regularization helps to control the influence of individual features in the model, preventing extreme coefficients.

To test different hyperparameter configurations in Logistic Regression, you can follow these steps:

1. Split your data into training and testing sets to evaluate the model's performance independently.

2. Define a range of hyperparameter values to test. For the regularization strength "C," you can create a list of values to try.

3. Use cross-validation techniques like k-fold cross-validation to evaluate each hyperparameter configuration. Cross-validation helps to get more reliable performance estimates and avoid overfitting during hyperparameter tuning.

4. Select the best hyperparameter configuration based on cross-validated performance. This could be the configuration that yields the highest accuracy or other relevant evaluation metrics.

5. Train the Logistic Regression model using the best hyperparameter configuration on the entire training dataset.

6. Evaluate the model's performance on the test set to get an estimate of its generalization ability.

By testing different hyperparameter configurations and selecting the best one based on cross-validation, you can build a Logistic Regression model that performs well on unseen data and achieves better generalization.

Advantages of Logistic Regression

- It is very simple and easy to implement.
- The output is more informative than other classification algorithms
- It expresses the relationship between independent and dependent variables
- Very effective with linearly separable data

Disadvantages of Logistic Regression

- Not effective with data that are not linearly separable
- Not as powerful as other classification models
- Multiclass classifications are much easier to do with other algorithms than logistic regression
- It can only predict categorical outcomes