**Agenda:**

* Understanding the Mathematical Logic of the Logistic Regression Algorithm.
* Parameters and Hyper-parameters in the Algorithm.
* Mathematical importance and logic behind each of those parameters and hyper-parameters.
* Applications of the Algorithm.

***“Understanding the Mathematical Logic of the Logistic Regression Algorithm”.***

**Introduction:**

Machine learning has revolutionized the world of business and is helping us build sophisticated applications to solve tough business problems. Using supervised and unsupervised machine learning models, you can solve problems using classification, regression, and clustering algorithms. Here, supervised machine-learning known as logistic regression in Python. Logistic regression can be used to solve both classification and regression problems.

**Introduction to Supervised Learning:**

Supervised machine learning algorithms derive insights, patterns, and relationships from a labelled training dataset. It means the dataset already contains a known value for the target variable for each record.  It is called supervised learning because the process of an algorithm learning from the training dataset is like an instructor supervising the learning process.

Supervised learning problems can be further classified into regression and classification problems.

* Classification: In a classification problem, the output variable is a category, such as “red” or “blue,” “disease” or “no disease,” “true” or “false,” etc.
* Regression: In a regression problem, the output variable is a real continuous value, such as “dollars” or “weight.”

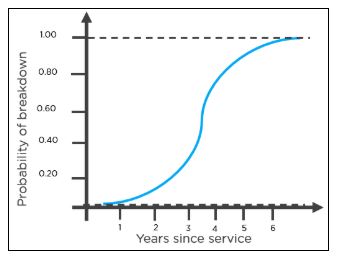
**What is Logistic Regression?**

Logistic regression is a statistical method that is used for building machine learning models where the dependent variable is dichotomous: i.e. binary. Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables. The independent variables can be nominal, ordinal, or of interval type.

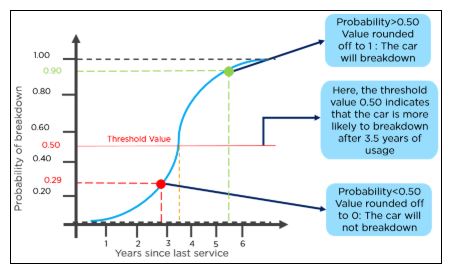
The name “logistic regression” is derived from the concept of the logistic function that it uses. The logistic function is also known as the sigmoid function. The value of this logistic function lies between zero and one.

**Example:**

The following is an example of a logistic function we can use to find the probability of a vehicle breaking down, depending on how many years it has been since it was serviced last.



Here is how you can interpret the results from the graph to decide whether the vehicle will break down or not.



Advantages of the Logistic Regression Algorithm

* Logistic regression performs better when the data is linearly separable
* It does not require too many computational resources as it’s highly interpretable
* There is no problem scaling the input features—It does not require tuning
* It is easy to implement and train a model using logistic regression
* It gives a measure of how relevant a predictor (coefficient size) is, and its direction of association (positive or negative)

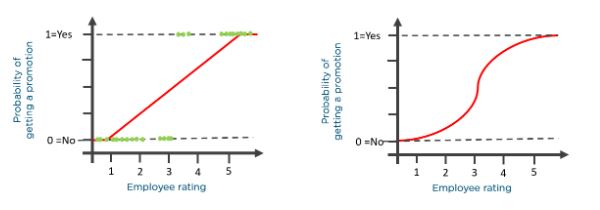
## How Does the Logistic Regression Algorithm Work?

Consider the following example: An organization wants to determine an employee’s salary increase based on their performance.

For this purpose, a linear regression algorithm will help them decide. Plotting a regression line by considering the employee’s performance as the independent variable, and the salary increase as the dependent variable will make their task easier.



Now, what if the organization wants to know whether an employee would get a promotion or not based on their performance? The above linear graph won’t be suitable in this case. As such, we clip the line at zero and one, and convert it into a sigmoid curve (S curve).



Based on the threshold values, the organization can decide whether an employee will get a salary increase or not.

To understand logistic regression, let’s go over the odds of success.

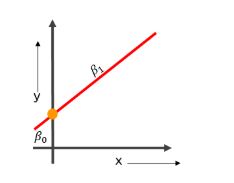
Odds (𝜃) = Probability of an event happening / Probability of an event not happening

𝜃 = p / 1 - p

The values of odds range from zero to ∞ and the values of probability lies between zero and one.

Consider the equation of a straight line:

𝑦 = 𝛽0 + 𝛽1\* 𝑥



Here, 𝛽0 is the y-intercept

𝛽1 is the slope of the line

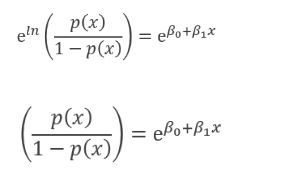
x is the value of the x coordinate

y is the value of the prediction

Now to predict the odds of success, we use the following formula:

log

Exponentiating both the sides, we have:



Let Y = e 𝛽0+𝛽1 \* 𝑥

Then p(x) / 1 - p(x) = Y

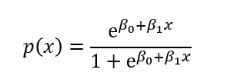
p(x) = Y(1 - p(x))

p(x) = Y - Y(p(x))

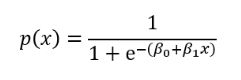
p(x) + Y(p(x)) = Y

p(x)(1+Y) = Y

p(x) = Y / 1+Y



The equation of the sigmoid function is:



The sigmoid curve obtained from the above equation is as follows:

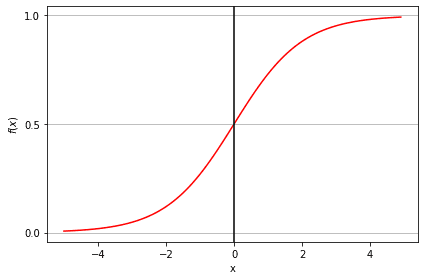


* “The log of the odds ratio is called logit, and the transformed model is linear in 𝛽’s”.

### Sigmoid Function

The sigmoid function, also called logistic function gives an ‘S’ shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO . For example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that a patient will suffer from cancer.





Linear Regression vs. Logistic Regression

|  |  |  |
| --- | --- | --- |
| ***Linear Regression*** | ***Logistic Regression*** |  |
| Used to solve regression problems | Used to solve classification problems |  |
| The response variables are continuous in nature | The response variable is categorical in nature |  |
| It helps estimate the dependent variable when there is a change in the independent variable | It helps to calculate the possibility of a particular event taking place |  |
| It is a straight line | It is an S-curve (S = Sigmoid) |  |

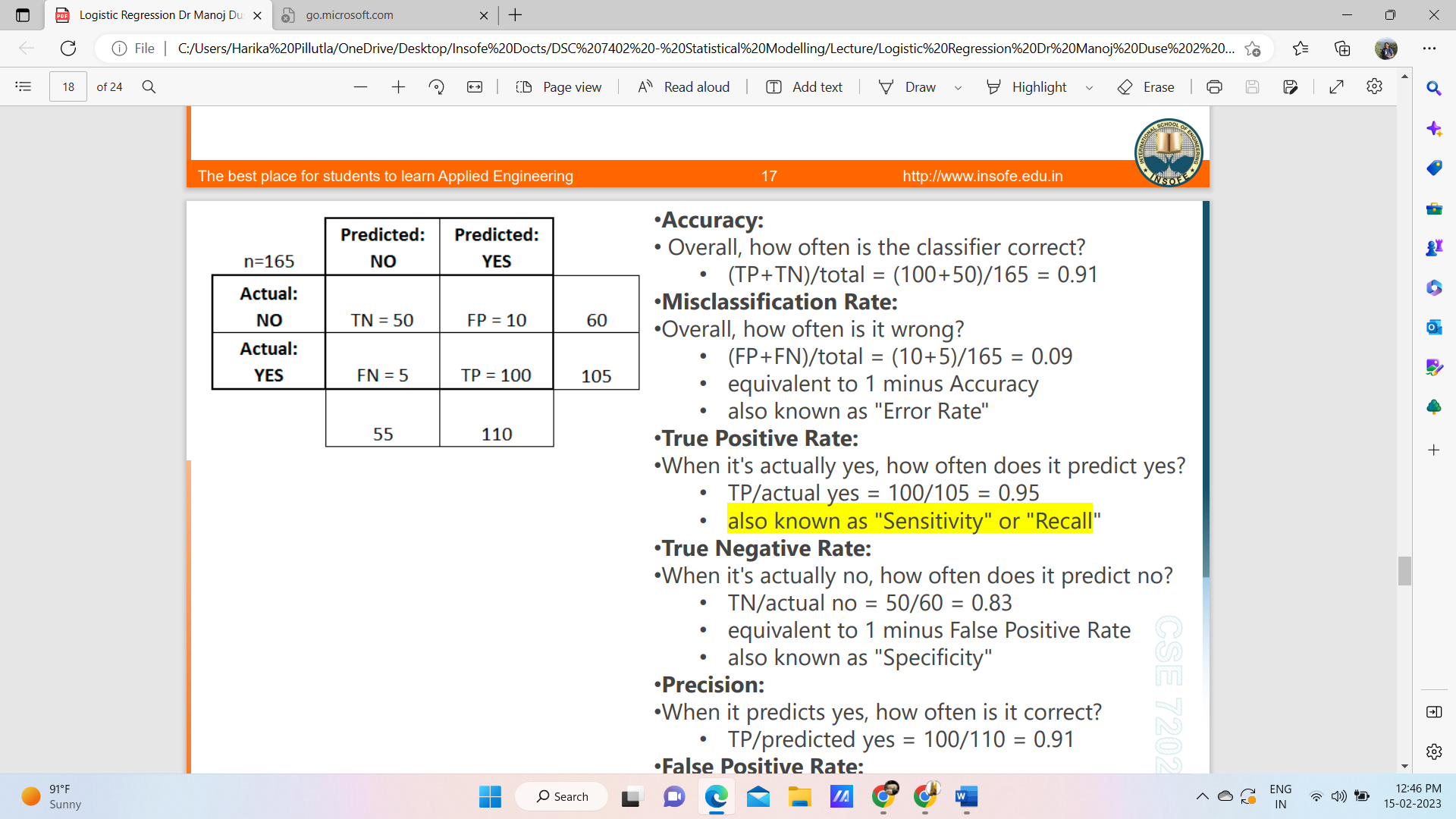
**There are two other types of logistic regression that depend on the number of predicted outcomes.**

### The three types of logistic regression

1. **Binary logistic regression** - When we have two possible outcomes, like our original example of whether a person is likely to be infected with COVID-19 or not.
2. **Multinomial logistic regression** - When we have multiple outcomes, say if we build out our original example to predict whether someone may have the flu, an allergy, a cold, or COVID-19.
3. **Ordinal logistic regression** - When the outcome is ordered, like if we build out our original example to also help determine the severity of a COVID-19 infection, sorting it into mild, moderate, and severe cases.

***“Parameters and Hyper-parameters in the Algorithm”.***

**Model Evaluation Using Confusion Matrix:**



**•Accuracy:**

• Overall, how often is the classifier correct?

• (TP+TN)/total = (100+50)/165 = 0.91

**•Misclassification Rate:**

•Overall, how often is it wrong?

• (FP+FN)/total = (10+5)/165 = 0.09

• equivalent to 1 minus Accuracy

• also known as "Error Rate"

**•True Positive Rate:**

•When it's actually yes, how often does it predict yes?

• TP/actual yes = 100/105 = 0.95

• also known as "Sensitivity" or "Recall"

**•True Negative Rate:**

•When it's actually no, how often does it predict no?

• TN/actual no = 50/60 = 0.83

• equivalent to 1 minus False Positive Rate

• also known as "Specificity"

**•Precision:**

•When it predicts yes, how often is it correct?

• TP/predicted yes = 100/110 = 0.91

**•False Positive Rate:**

•When it's actually no, how often does it predict yes?

• FP/actual no = 10/60 = 0.17

**ROC Curves and AUC**:

• ROC – Receiver Operating Characteristics

• AUC – Area Under the ROC Curve

• At a given threshold, we can evaluate the classification accuracy (accuracy, sensitivity, recall, etc)

• ROC curve tries to evaluate how well the regression has achieved the separation between the classes at all threshold values.

**ROC Curves and AUC**

Rough rule of thumb:

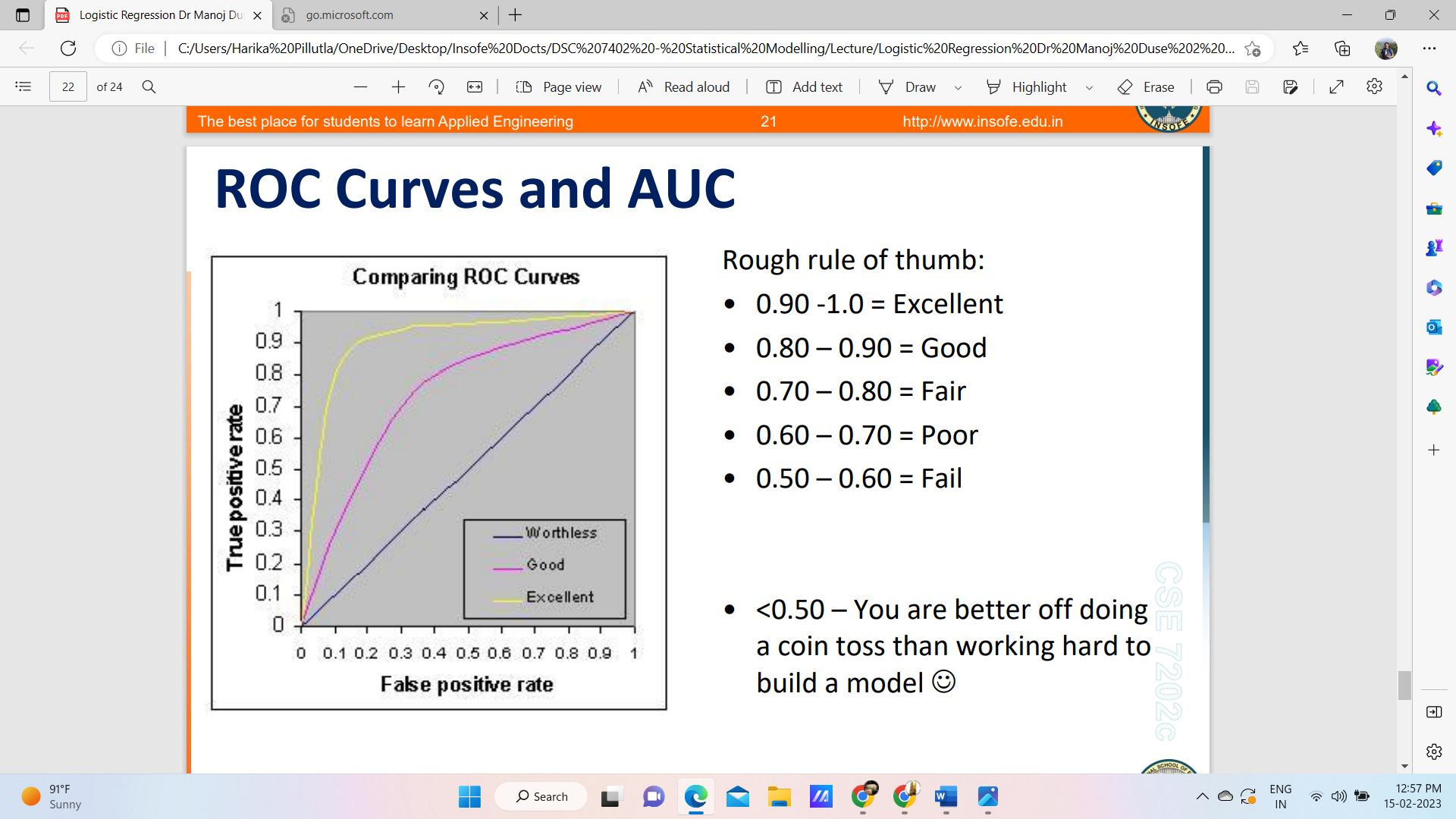
• 0.90 -1.0 = Excellent

• 0.80 – 0.90 = Good

• 0.70 – 0.80 = Fair

• 0.60 – 0.70 = Poor

• 0.50 – 0.60 = Fail



***“Mathematical importance and logic behind each of those parameters and hyper-parameters.”***

**Parameters *:***

Following table lists the parameters used by **Logistic Regression** module −

|  |  |
| --- | --- |
| **Sr.No** | **Parameter & Description** |
| 1 | ***penalty****− str, ‘L1’, ‘L2’, ‘elasticnet’ or none, optional, default = ‘L2’*  This parameter is used to specify the norm (L1 or L2) used in penalization (regularization). |
| 2 | ***dual****− Boolean, optional, default = False*  It is used for dual or primal formulation whereas dual formulation is only implemented for L2 penalty. |
| 3 | ***tol****− float, optional, default=1e-4*  It represents the tolerance for stopping criteria. |
| 4 | ***C****− float, optional, default=1.0*  It represents the inverse of regularization strength, which must always be a positive float. |
| 5 | ***fit\_intercept****− Boolean, optional, default = True*  This parameter specifies that a constant (bias or intercept) should be added to the decision function. |
| 6 | ***intercept\_scaling****− float, optional, default = 1*  This parameter is useful when   * the **solver ‘liblinear’** is used * ***fit\_intercept*** is set to true |
| 7 | ***class\_weight****− dict or ‘balanced’ optional, default = none*  It represents the weights associated with classes. If we use the default option, it means all the classes are supposed to have weight one. On the other hand, if you choose class\_weight: balanced, it will use the values of y to automatically adjust weights. |
| 8 | ***random\_state****− int, RandomState instance or None, optional, default = none*  This parameter represents the seed of the pseudo random number generated which is used while shuffling the data. Followings are the options   * **int** − in this case, random\_state is the seed used by random number generator. * **RandomState instance** − in this case, *random\_state* is the random number generator. * **None** − in this case, the random number generator is the RandonState instance used by np.random. |
| 9 | ***solver****− str, {‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘saag’, ‘saga’}, optional, default = ‘liblinear’*  This parameter represents which algorithm to use in the optimization problem. Followings are the properties of options under this parameter −   * **liblinear** − It is a good choice for small datasets. It also handles L1 penalty. For multiclass problems, it is limited to one-versus-rest schemes. * **newton-cg** − It handles only L2 penalty. * **lbfgs** − For multiclass problems, it handles multinomial loss. It also handles only L2 penalty. * **saga** − It is a good choice for large datasets. For multiclass problems, it also handles multinomial loss. Along with L1 penalty, it also supports ‘elasticnet’ penalty. * **sag** − It is also used for large datasets. For multiclass problems, it also handles multinomial loss. |
| 10 | ***max\_iter****− int, optional, default = 100*  As name suggest, it represents the maximum number of iterations taken for solvers to converge. |
| 11 | ***multi\_class****− str, {‘ovr’, ‘multinomial’, ‘auto’}, optional, default = ‘ovr’*   * **ovr** − For this option, a binary problem is fit for each label. * **multimonial** − For this option, the loss minimized is the multinomial loss fit across the entire probability distribution. We can’t use this option if solver = ‘liblinear’. * **auto** − This option will select ‘ovr’ if solver = ‘liblinear’ or data is binary, else it will choose ‘multinomial’. |
| 12 | ***verbose****− int, optional, default = 0*  By default, the value of this parameter is 0 but for liblinear and lbfgs solver we should set verbose to any positive number. |
| 13 | ***warm\_start****− bool, optional, default = false*  With this parameter set to True, we can reuse the solution of the previous call to fit as initialization. If we choose default i.e. false, it will erase the previous solution. |
| 14 | ***n\_jobs****− int or None, optional, default = None*  If multi\_class = ‘ovr’, this parameter represents the number of CPU cores used when parallelizing over classes. It is ignored when solver = ‘liblinear’. |
| 15 | ***l1\_ratio****− float or None, optional, dgtefault = None*  It is used in case when penalty = ‘elasticnet’. It is basically the Elastic-Net mixing parameter with 0 < = l1\_ratio > = 1. |

**Attributes*:*** Followings table consist the attributes used by **Logistic Regression** module −

|  |  |
| --- | --- |
| **Sr.No** | **Attributes & Description** |
| 1 | ***coef\_****− array, shape(n\_features,) or (n\_classes, n\_features)*  It is used to estimate the coefficients of the features in the decision function. When the given problem is binary, it is of the shape (1, n\_features). |
| 2 | ***Intercept\_****− array, shape(1) or (n\_classes)*  It represents the constant, also known as bias, added to the decision function. |
| 3 | ***classes\_****− array, shape(n\_classes)*  It will provide a list of class labels known to the classifier. |
| 4 | ***n\_iter\_****− array, shape (n\_classes) or (1)*  It returns the actual number of iterations for all the classes. |

***“Applications of the Algorithm”:***



* Using the logistic regression algorithm, banks can predict whether a customer would default on loans or not
* To predict the weather conditions of a certain place (sunny, windy, rainy, humid, etc.)
* Ecommerce companies can identify buyers if they are likely to purchase a certain product
* Companies can predict whether they will gain or lose money in the next quarter, year, or month based on their current performance
* To classify objects based on their features and attributes.

Assumption in a Logistic Regression Algorithm:

* In a binary logistic regression, the dependent variable must be binary
* Only meaningful variables should be included
* The independent variables should be independent of each other. This means the model should have little or no multicollinearity
* The independent variables are linearly related to the log odds
* Logistic regression requires quite large sample sizes.

**Refrences:**

<https://www.capitalone.com/tech/machine-learning/what-is-logistic-regression/>

<https://www.tutorialspoint.com/scikit_learn/scikit_learn_logistic_regression.htm>

<https://www.simplilearn.com/tutorials/machine-learning-tutorial/logistic-regression-in-python>