**What is Logistic Regression?**

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurrence.

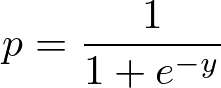
It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

**Linear Regression Equation:**

eq1

Where, y is a dependent variable and x1, x2 ... and Xn are explanatory variables.

**Sigmoid Function:**



**Apply Sigmoid function on linear regression:**

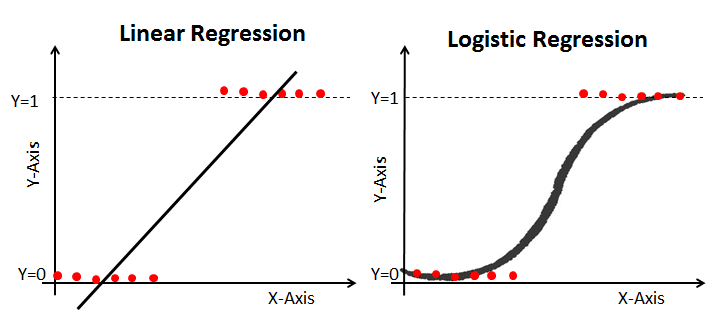


**Properties of Logistic Regression:**

* The dependent variable in logistic regression follows Bernoulli Distribution.
* Estimation is done through maximum likelihood.
* No R Square, Model fitness is calculated through Concordance, KS-Statistics.

**Linear Regression Vs. Logistic Regression**

[**Linear regression**](https://www.datacamp.com/tutorial/essentials-linear-regression-python) gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.



**Maximum Likelihood Estimation Vs. Least Square Method**

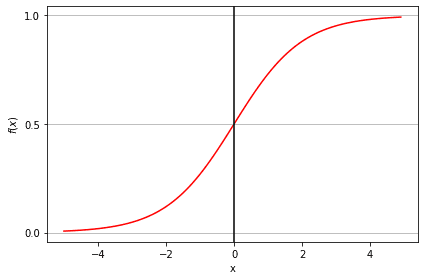
The MLE is a "likelihood" maximization method, while OLS is a distance-minimizing approximation method. Maximizing the likelihood function determines the parameters that are most likely to produce the observed data. From a statistical point of view, MLE sets the mean and variance as parameters in determining the specific parametric values for a given model. This set of parameters can be used for predicting the data needed in a normal distribution.

Ordinary Least squares estimates are computed by fitting a regression line on given data points that has the minimum sum of the squared deviations (least square error). Both are used to estimate the parameters of a linear regression model. MLE assumes a joint probability mass function, while OLS doesn't require any stochastic assumptions for minimizing distance.

**Sigmoid Function**

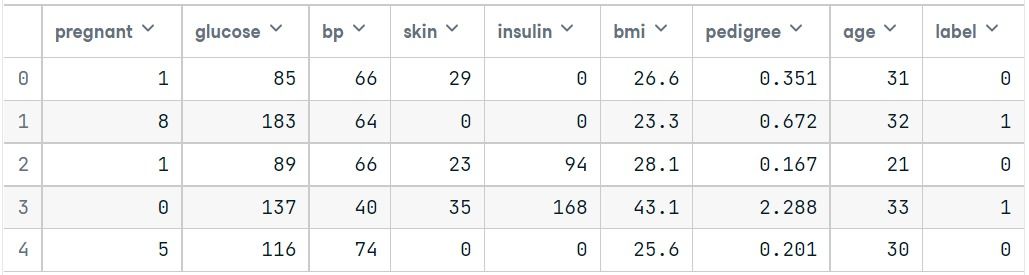
The sigmoid function, also called logistic function gives an ‘S’ shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the 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. The outputcannotFor 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.





**Types of Logistic Regression**

Types of Logistic Regression:

* Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
* Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
* Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.
* **Model building in Scikit-learn**
* Let's build the diabetes prediction model.
* Here, you are going to predict diabetes using the Logistic Regression Classifier.
* Let's first load the required Pima Indian Diabetes dataset using the pandas' read CSV function. You can download data from the following link: [**https://www.kaggle.com/uciml/pima-indians-diabetes-database**](https://www.kaggle.com/uciml/pima-indians-diabetes-database) or select a dataset from DataCamp: [**https://www.datacamp.com/workspace/datasets**](https://www.datacamp.com/workspace/datasets). The ready-to-use dataset provides you the option to train the model on **[DataCamp's Workspace](https://www.datacamp.com/workspace" \t "_blank)**, which is a free Jupyter notebook on the cloud.
* **Loading Data**
* We will simplify columns by providing col\_names to pandas read\_csv() function.
* #import pandas
* import pandas as pd
* col\_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
* # load dataset
* pima = pd.read\_csv("pima-indians-diabetes.csv", header=None, names=col\_names)
* OpenAI
* pima.head()
* 

### Selecting Feature

Here, you need to divide the given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

#split dataset in features and target variable

feature\_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']

X = pima[feature\_cols] # Features

y = pima.label # Target variable

OpenAI

### Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split the dataset by using the function train\_test\_split(). You need to pass 3 parameters: features, target, and test\_set size. Additionally, you can use random\_state to select records randomly.

# split X and y into training and testing sets

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=16)

OpenAI

Here, the Dataset is broken into two parts in a ratio of 75:25. It means 75% data will be used for model training and 25% for model testing.

### Model Development and Prediction

First, import the Logistic Regression module and create a Logistic Regression classifier object using the LogisticRegression() function with random\_state for reproducibility.

Then, fit your model on the train set using fit() and perform prediction on the test set using predict().

# import the class

from sklearn.linear\_model import LogisticRegression

# instantiate the model (using the default parameters)

logreg = LogisticRegression(random\_state=16)

# fit the model with data

logreg.fit(X\_train, y\_train)

y\_pred = logreg.predict(X\_test)

OpenAI

## Model Evaluation using Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. You can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions summed up class-wise.

# import the metrics class

from sklearn import metrics

cnf\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)

cnf\_matrix

OpenAI

array([[115, 8],

[ 30, 39]])

OpenAI

Here, you can see the confusion matrix in the form of the array object. The dimension of this matrix is 2\*2 because this model is binary classification. You have two classes 0 and 1. Diagonal values represent accurate predictions, while non-diagonal elements are inaccurate predictions. In the output, 115 and 39 are actual predictions, and 30 and 8 are incorrect predictions.

### Visualizing Confusion Matrix using Heatmap

Let's visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn.

Here, you will visualize the confusion matrix using Heatmap.

# import required modules

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

class\_names=[0,1] # name of classes

fig, ax = plt.subplots()

tick\_marks = np.arange(len(class\_names))

plt.xticks(tick\_marks, class\_names)

plt.yticks(tick\_marks, class\_names)

# create heatmap

sns.heatmap(pd.DataFrame(cnf\_matrix), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual label')

plt.xlabel('Predicted label')

Text(0.5,257.44,'Predicted label');

OpenAI

### cm

### Confusion Matrix Evaluation Metrics

Let's evaluate the model using classification\_report for accuracy, precision, and recall.

from sklearn.metrics import classification\_report

target\_names = ['without diabetes', 'with diabetes']

print(classification\_report(y\_test, y\_pred, target\_names=target\_names))

OpenAI

precision recall f1-score support

without diabetes 0.79 0.93 0.86 123

with diabetes 0.83 0.57 0.67 69

accuracy 0.80 192

macro avg 0.81 0.75 0.77 192

weighted avg 0.81 0.80 0.79 192

OpenAI

Well, you got a classification rate of **80%**, considered as good accuracy.

**Precision**: Precision is about being precise, i.e., how accurate your model is. In other words, you can say, when a model makes a prediction, how often it is correct. In your prediction case, when your Logistic Regression model predicted patients are going to suffer from diabetes, that patients have 73% of the time.

**Recall**: If there are patients who have diabetes in the test set and your Logistic Regression model can identify it 57% of the time.

### ROC Curve

Receiver Operating Characteristic(ROC) curve is a plot of the true positive rate against the false positive rate. It shows the tradeoff between sensitivity and specificity.

y\_pred\_proba = logreg.predict\_proba(X\_test)[::,1]

fpr, tpr, \_ = metrics.roc\_curve(y\_test, y\_pred\_proba)

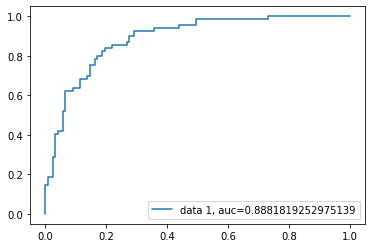
auc = metrics.roc\_auc\_score(y\_test, y\_pred\_proba)

plt.plot(fpr,tpr,label="data 1, auc="+str(auc))

plt.legend(loc=4)

plt.show()

OpenAI



AUC score for the case is 0.88. AUC score 1 represents a perfect classifier, and 0.5 represents a worthless classifier.

The code source is available at Workspace: [**Understanding Logistic Regression in Python**](https://app.datacamp.com/workspace/w/11b0dd9d-49be-429d-992e-45f73b3c23fb).

## Advantages

Because of its efficient and straightforward nature, it doesn't require high computation power, is easy to implement, easily interpretable, and used widely by data analysts and scientists. Also, it doesn't require scaling of features. Logistic regression provides a probability score for observations.

## Disadvantages

Logistic regression is not able to handle a large number of categorical features/variables. It is vulnerable to overfitting. Also, can't solve the non-linear problem with the logistic regression that is why it requires a transformation of non-linear features. Logistic regression will not perform well with independent variables that are not correlated to the target variable and are very similar or correlated to each other.