AI-ML Module 1: Logistic Regression

1. Logistic Regression

1.1 Start

Notes:

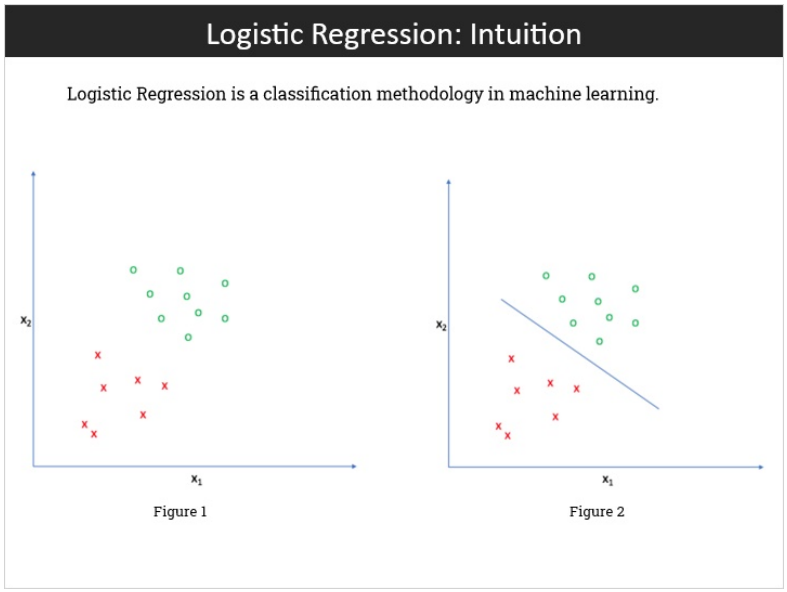
In this chapter we shall learn about Logistic Regression.

It is recommended to complete previous chapters before going further for seamless understanding.

Transcript can be downloaded from resources.

Happy Learning!

1.2 Intuition



Notes:

Logistic regression is a classification methodology in machine learning. Let us understand it with the help of an example. On the slide, You can see some points in a 2 dimensional space. The dimensions or features are named X 1 and X 2. There are two types of points. The points in the form of green circles are of one type which we shall call 0 or positive. The points in the form of red crosses are of other type, and we shall call these as 1 or negative. We can see that the 2 types of points are well separated. However, we need a representation or a model to separate the 2 types of points. A line separating the 2 types of points is shown on the slide. This is an example of linear model for classification. The example considered here is for binary classification, where there are two types of data points. In machine learning binary classification problems, two categories are represented as 0 or 1 for mathematical calculations. The categories are generally named High or Low, True or False, Positive or Negative.

1.3 Intuition

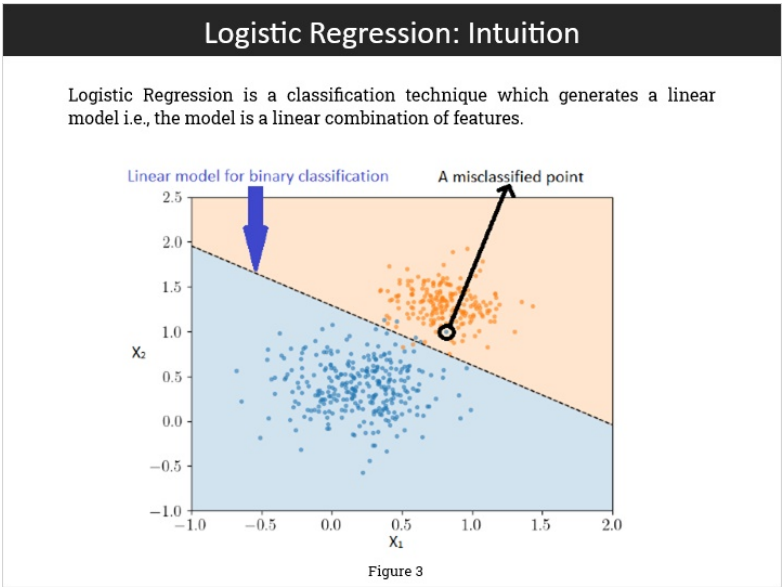
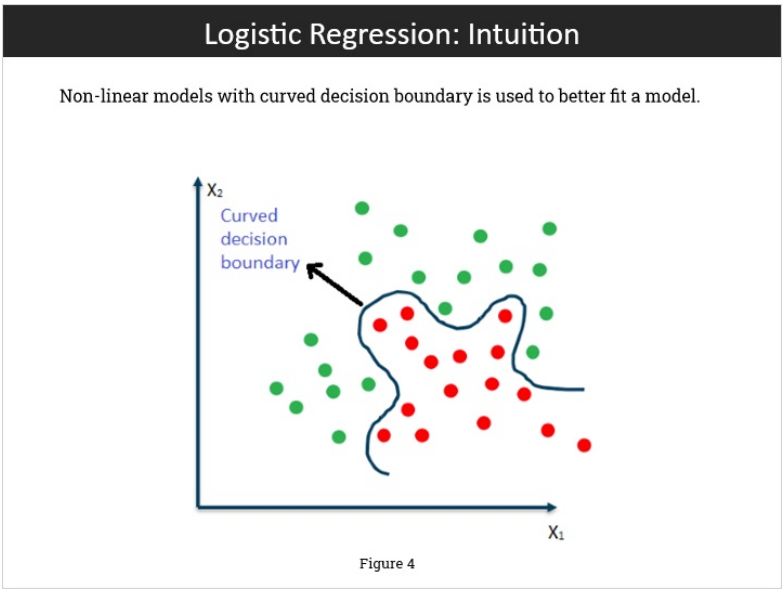


Figure 3

Notes:

Logistic Regression is a classification technique which uses a model similar to that indicated in previous slide. A linear model means the model is a linear combination of features and can be a line in 2 dimensions, a plane in 3 dimensions and a hyperplane in higher dimensions. Another example of logistic regression in action is shown on the slide. In this example a linear model for binary-classification is shown. The task in logistic regression is to find this model or line. This line is also known as decision boundary or decision surface as it decides or predicts the category of a point. In this image we can see that some points are on the wrong side of line that is they are misclassified. This misclassification depends upon accuracy of model. The higher the accuracy, the lower will be misclassification and the better will be model. We shall know about accuracy in detail as we proceed further.

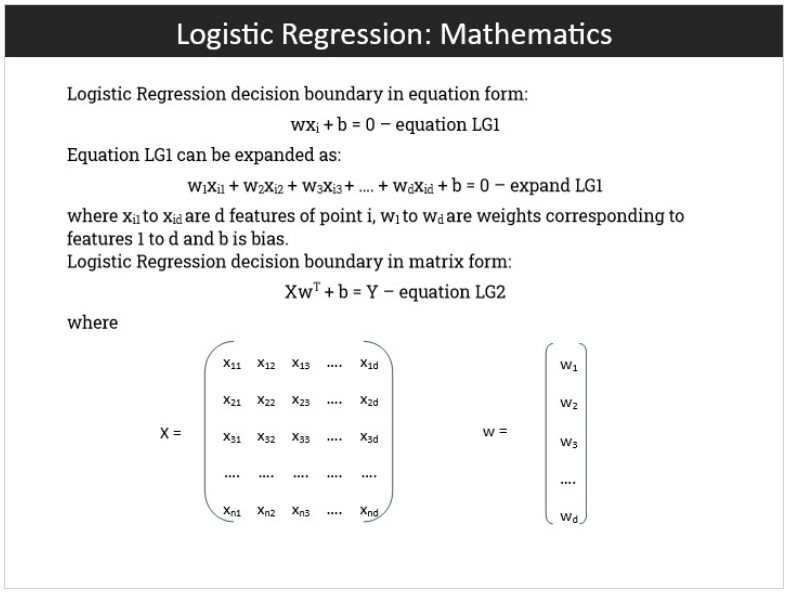
1.4 Intuition



Notes:

In this slide you will get a glimpse of non linear models. In order to get better decision boundary, Non linear models can also be adopted which curve around points to fit better. An example is shown on the slide. However, with the introduction of non-linearity and further increase in model complexity there are chances of over-fitting. So, bias-variance trade-off plays an important role in deciding the level of complexity.

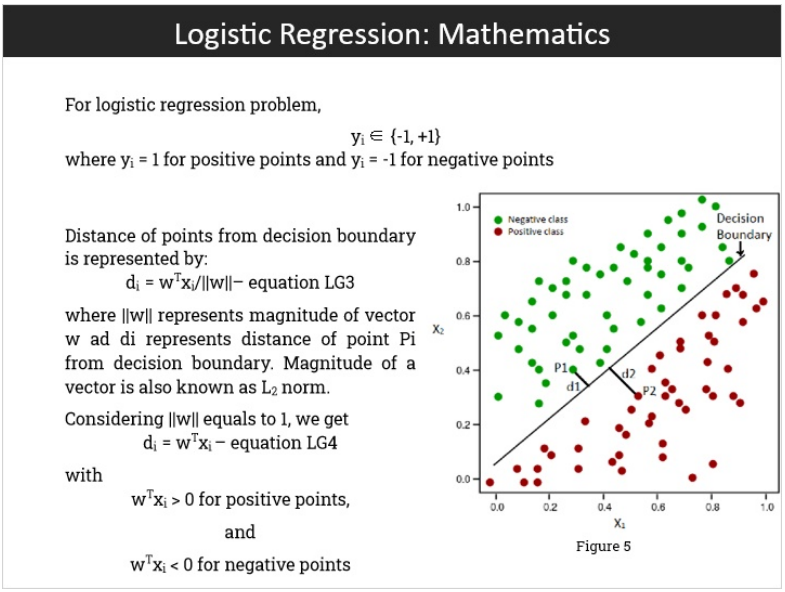
1.5 Mathematics



Notes:

Like linear regression, logistic regression is also a linear model. The decision boundary obtained is a linear combination of features and can be represented as equation LG1. Equation LG1 when expanded is represented as expand LG1. In matrix form decision boundary can be represented as equation LG2.

1.6 Mathematics



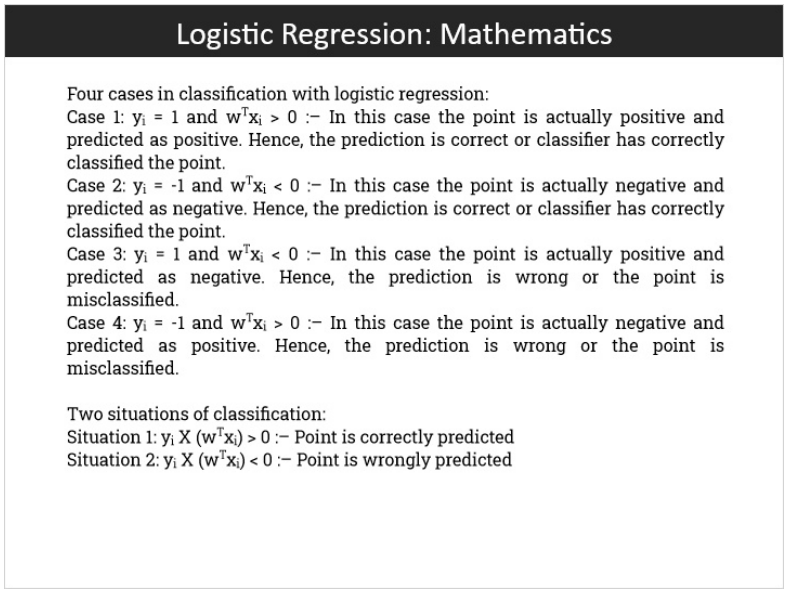
Notes:

We saw the equation of decision boundary in last slide. Now, let us understand how this decision boundary works for classification. Figure 5 on the slide shows a decision boundary classifying positive and negative points. This figure represents a binary classification problem. A sample positive point P1 is at a distance d1 from decision boundary. Similarly a sample negative point P2 is at a distance d2 from decision boundary.

For such problems, output for a point P is represented by y i, where y i can be either minus 1 or 1. When y i equals 1, it means the point is positive. Similarly when y i equals minus 1, it means the point is negative. Thus for logistic regression, y i can be either 1 or minus 1.

From geometry and coordinate system we know that distance d of any point P from decision boundary which is a line in this case can be represented by equation LG3. Please note that for simplification we have ignored the bias term b in equation LG3. For ease of understanding let us assume magnitude of weight vector w is 1. We know that for a point lying on decision boundary, distance d of the point from line is 0, that is equation LG3 calculates to 0. For points not lying on decision boundary, equation LG3 will evaluate to either less than 0 or more than 0. Thus for a point predicted positive, d i will be positive, and for a point predicted negative, d i will be negative.

1.7 Mathematics

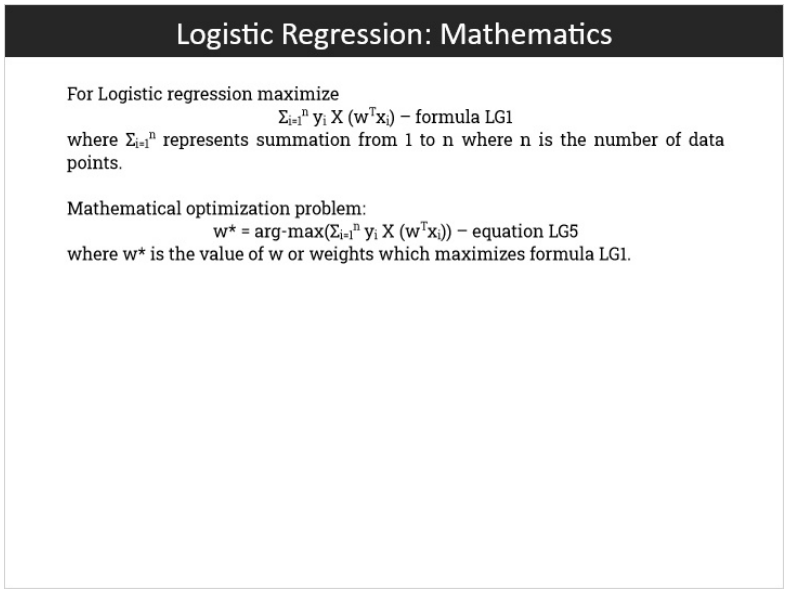


Notes:

Four different cases are possible in classification with logistic regression. These are listed on the slide. Please go through these cases carefully.

It can be noted that for correctly predicted cases or points, we have situation 1. And for wrongly predicted cases or points, we have situation 2. These situations are mentioned on the slide. For classifier to be good, as many as possible points should fall under situation 1.

1.8 Mathematics



Notes:

Thus in Logistic Regression, we need to maximize formula LG1. Formula LG1 represents summation of multiplication of y i and d i for all the data points. Distance d i is represented by equation LG4 on earlier slide. Small n in summation represents the total number of data points.

The mathematical optimization problem for Logistic regression can be represented as equation LG5. Here w star represents that value of w or weights which maximizes formula LG1.

The distance d i of a point i from decision boundary and represented by equation LG4, can take any positive or negative value. The value of d i can vary from very small to very large. In order to limit the vale of d i, sigmoid function is used on formula LG1 and thus the mathematical optimization function gets modified.

1.9 Mathematics

Logistic Regression: Mathematics

Sigmoid of x is represented by:

$$\sigma(x) = 1/(1+e^{-x}) \text{ – equation LG6}$$

Logistic regression aims to maximize z represented by the following equation:

$$z = \sigma(\Sigma_{i=1}^{n} y_i X (w^T x_i)) \text{ – equation LG7}$$

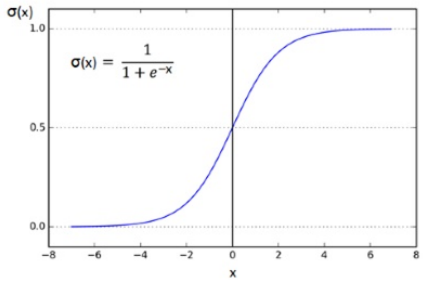


Figure 6

Mathematical optimization problem for Logistic Regression now becomes:

$$w^* = \text{arg-max}(\sigma(\Sigma_{i=1}^{n} y_i X (w^T x_i))) \text{ – equation LG8}$$

where w* is the value of w or weights which maximizes equation LG7.

Notes:

A sigmoid function is defined by equation LG6 on the slide. This equation defines sigmoid for a variable x. Sigmoid is used as squashing technique to limit the output value between 0 and 1. Thus upon applying sigmoid function, a small value remains small and a large value is reduced to a small value. Graphical representation of sigmoid is given by figure 6 on the slide. From figure 6, we can observe that sigmoid function limits the output value between 0 and 1 for any value of x. It can also be seen that output value lies between 0 and 0.5 for negative values of x and between 0.5 and 1 for positive values of x.

When this sigmoid squashing is applied on formula LG1 we get to maximize equation LG7 and get LG8 as optimization problem.

1.10 Mathematics

Logistic Regression: Mathematics

Mathematical optimization problem with regularization for Logistic Regression:

$$w^* = \arg\max(\sigma(\Sigma_{i=1}^{n} y_i X (w^T x_i)) + \text{Regularization}) - \text{equation LG9}$$

where w^* is the value of w or weights which maximizes equation LG7 plus regularization term.

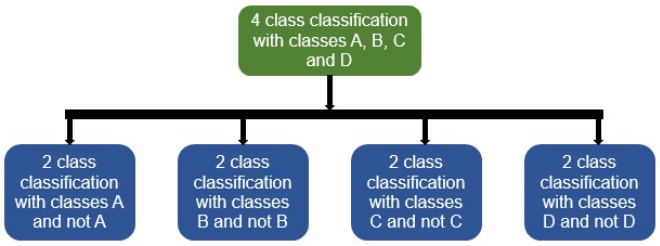
Notes:

In order to take care of overfitting, a regularization term is added in equation LG8. This in turn will modify the mathematical optimization problem as equation LG9. Finally gradient descent is used to get the weight values and thus the decision boundary.

1.11 Mathematics

Logistic Regression: Mathematics

Logistic Regression is inherently a binary (two-class) classifier which can be used for multi-class classification by using one versus rest approach among different classes.



Logistic Regression is a linear classifier giving linear decision boundaries. However, non-linearity can be introduced by using mathematical transformation on features. Trigonometric functions and polynomial functions are example of such transformation.

Notes:

A point to be noted is that logistic regression is inherently binary classifier or two class classifier. For multi-class classification, logistic regression is used as one versus rest classifier for each class. Let us understand this is detail. Suppose we have a multi class classification problem where we have four classes namely, A, B, C and D. This four class classification problem will be segregated into 4 binary classification problems. The 4 binary classification problems will be one versus rest in nature. One case of classification will have two classes, A and not A. Here not A will consist of B, C and D classes. Similarly other cases will be, B and not B, C and not C and lastly D and not D.

The point is that logistic regression is binary classifier in nature but can be used for multi class classification.

Another point to be noted for logistic regression is that the classifier or decision boundary is linear in nature, that is it is a linear combination of features. However, non linear classifier can be obtained by using mathematical techniques such as trigonometric functions, polynomials etc. For example if a feature called x 1 is converted into x 1 cube or sine x, it will lead to non-linearity of classifier or decision boundary. In mathematical language this is a case of transformation from one set of features or dimension space to other.

1.12 Mathematics

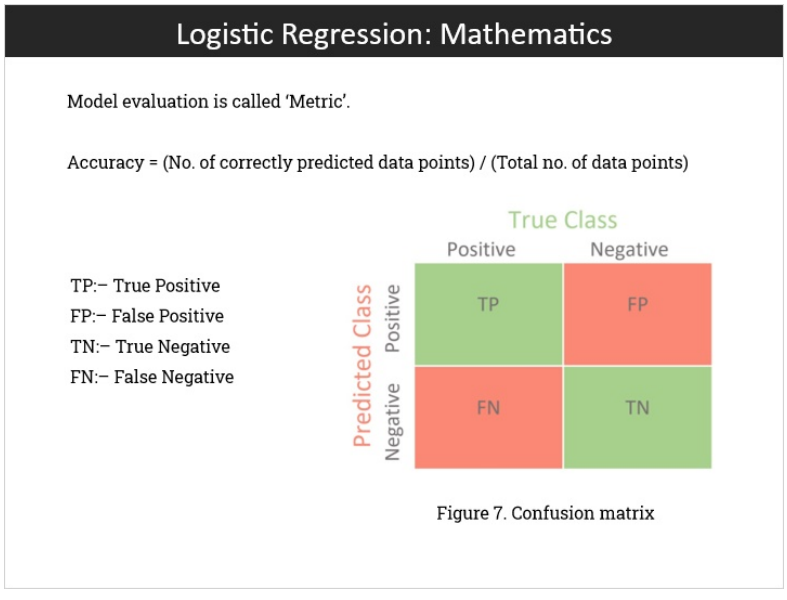


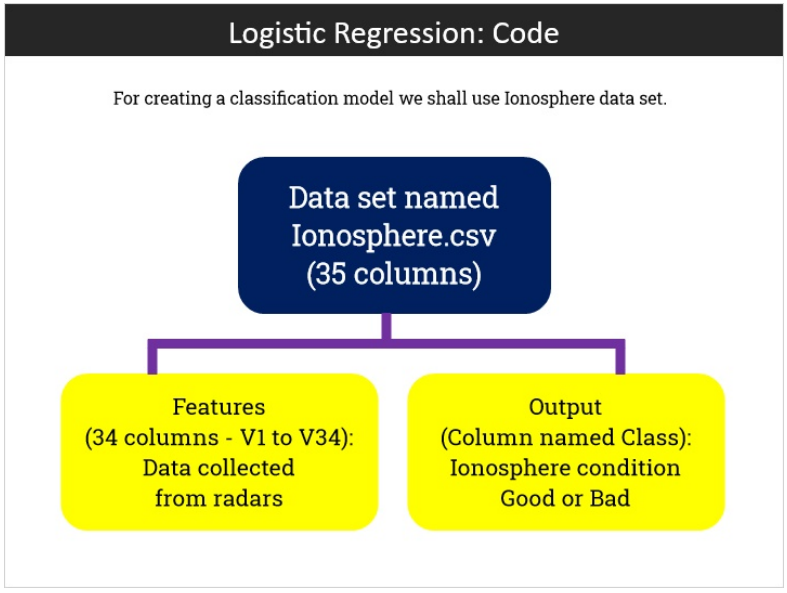
Figure 7. Confusion matrix

Notes:

We shall now discuss how to evaluate a classification model. The evaluation parameter is called metric. A very commonly used metric is accuracy. Accuracy is total number of correctly classified data points divided by total number of data points. There are other metrics for measuring performance of a model which will be discussed as and when required.

At this point let us understand a very useful representation of actual and predicted values. This representation is known as Confusion matrix. A confusion matrix is shown as figure 7. On the vertical axis we have count of predicted class and on the horizontal axis we have count of true class. The classes shown in figure 7 are positive and negative. However, the classes may be 0 and 1 or any other combination as defined. The matrix is divided into four quadrants showing the count of true positive, false positive, true negative and false negative. True positive shows count of data points which are predicted as positive and are actually positive. The names of all the four quadrants can be split into two words. First word represents whether the prediction is true or false that is correct or wrong. The second word represents the predicted value, that is positive or negative in this case. Thus for True Positive points, true represents that the points are correctly predicted and positive represents that the points are predicted as positive. Similar is the case with other quadrants.

1.13 Code

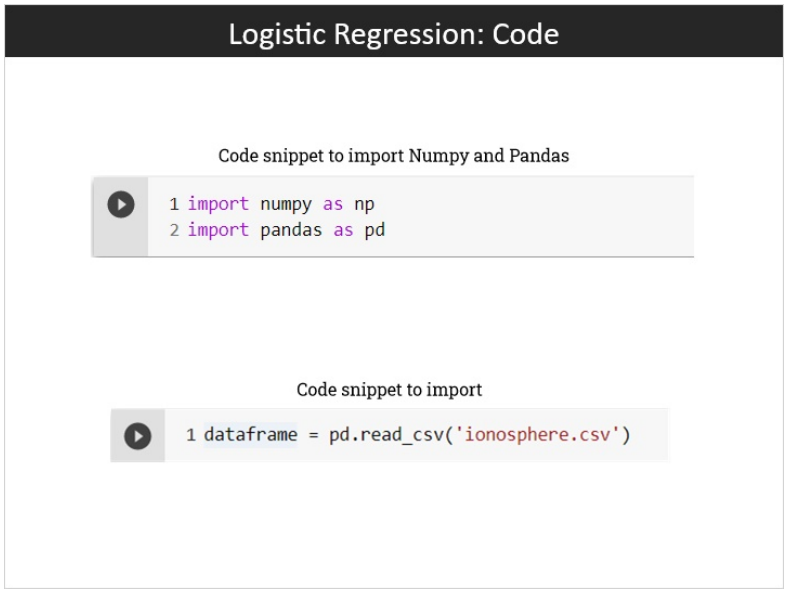


Notes:

Now, we shall write code to create a logistic regression model on ionosphere data set.

Code file in the form of dot ipynb file & dot py file and data set can be downloaded from resources. The ionosphere data set in the form of a c s v file consists of 35 columns. Columns named V 1 to V 34 are features which are data collected from radars. Column named class indicates whether condition of ionosphere is good or bad. This data set is a clean data set and does not need preprocessing. However, we have to convert the current class values into numerical values. Let us start coding using Colab.

1.14 Code

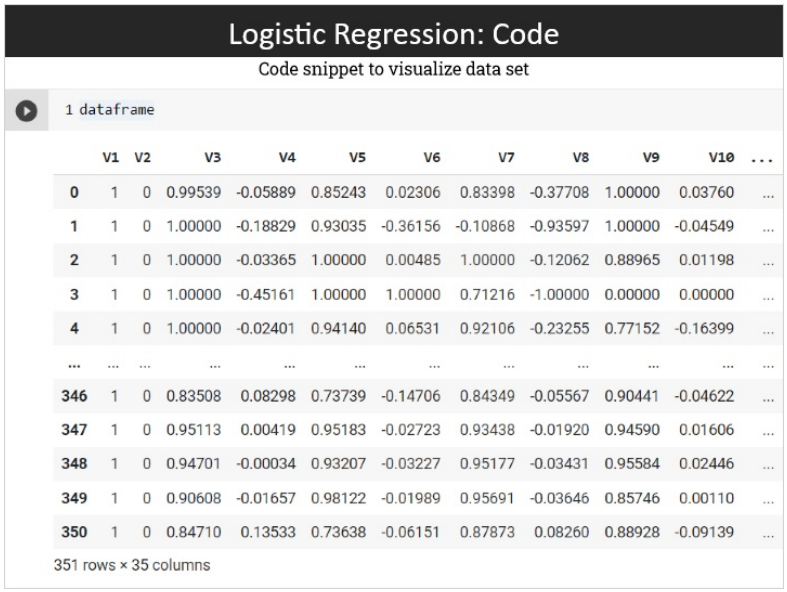


Notes:

In the first code snippet on this slide, some useful packages namely numpy and pandas are imported.

Next code snippet is used to import the data set and save it as a data frame in the name of a variable named data frame.

1.15 Code



Logistic Regression: Code

Code snippet to visualize data set

```
1 dataframe
```

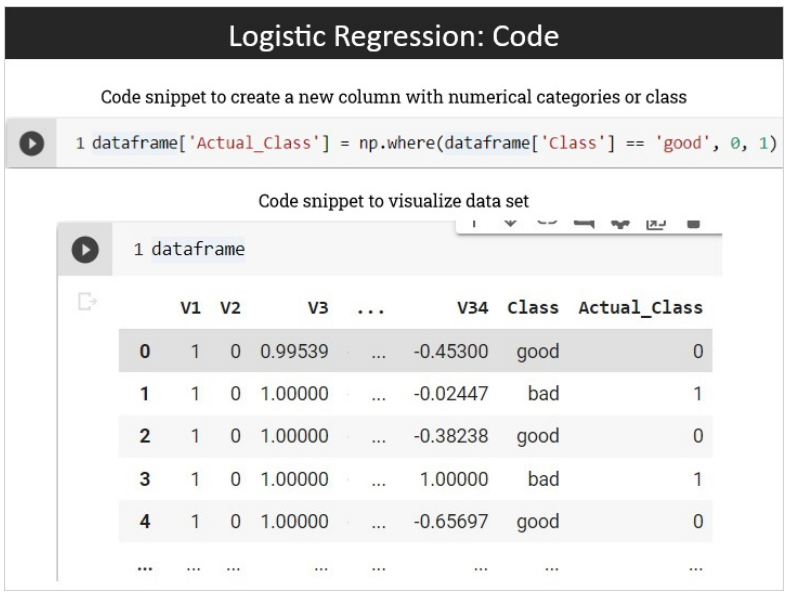
	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...
0	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	...
1	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...
2	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...
3	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...
4	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...
...	...	...	...	...	...	...	...	...	...	...	...
346	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...
347	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...
348	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...
349	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...
350	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...

351 rows × 35 columns

Notes:

On this slide, code snippet for visualization of data set is shown. We can see that there are 351 rows or data points and 35 columns.

1.16 Code



Logistic Regression: Code

Code snippet to create a new column with numerical categories or class

```
1 dataframe['Actual_Class'] = np.where(dataframe['Class'] == 'good', 0, 1)
```

Code snippet to visualize data set

```
1 dataframe
```

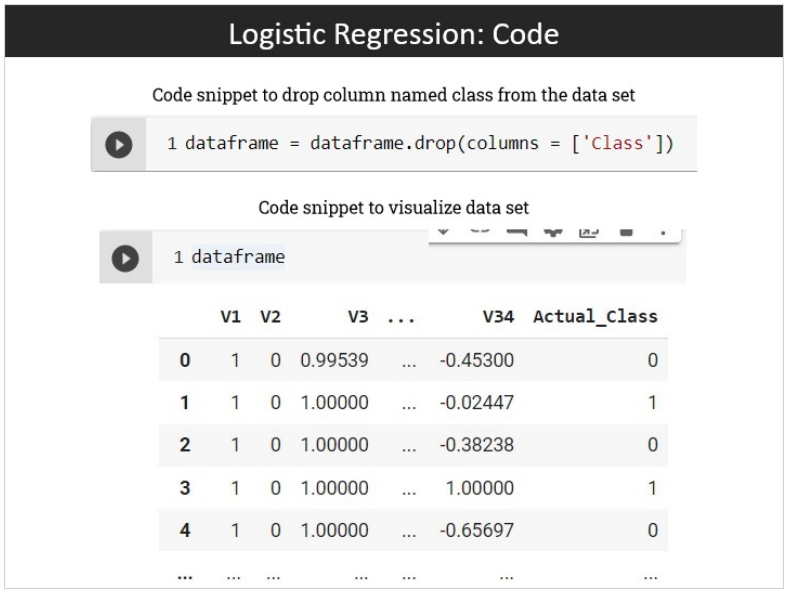
	V1	V2	V3	...	V34	Class	Actual_Class
0	1	0	0.99539	...	-0.45300	good	0
1	1	0	1.00000	...	-0.02447	bad	1
2	1	0	1.00000	...	-0.38238	good	0
3	1	0	1.00000	...	1.00000	bad	1
4	1	0	1.00000	...	-0.65697	good	0
...	...	...	...	...	...	...	...

Notes:

The first code snippet on this slide is used to create a new column named actual underscore class. In this column a data point is assigned a value of 0 if the corresponding class value is good and 1 if the corresponding class value is bad. This new column creates numerical categories 0 and 1 for the data points.

Next code snippet is used to visualize the data set with new column. We can see a new column named actual underscore class with categories 0 and 1 for each data point.

1.17 Code



	V1	V2	V3	...	V34	Actual_Class
0	1	0	0.99539	...	-0.45300	0
1	1	0	1.00000	...	-0.02447	1
2	1	0	1.00000	...	-0.38238	0
3	1	0	1.00000	...	1.00000	1
4	1	0	1.00000	...	-0.65697	0
...	...	...	...	...	...	...

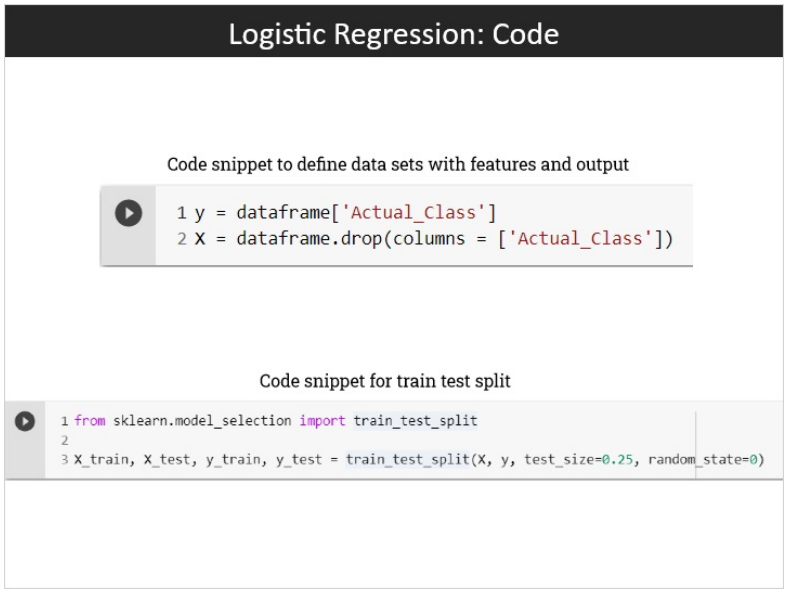
Notes:

The first code snippet on this slide is used to drop the column named Class from the data set. The class column is dropped as we have a new column named actual underscore class for numerical categorization of data points.

Next code snippet is used to visualize the new data set. We can see that column named class is dropped.

The required manipulation in the data set is now complete.

1.18 Code

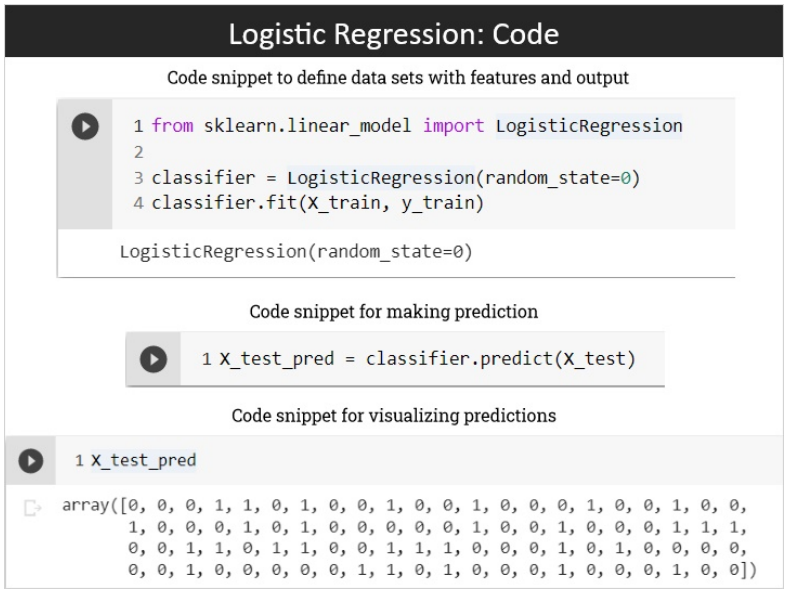


Notes:

In the first code snippet, 2 data frames are created. One consists of features and named X. Other consists of output or categories and is named y.

In the next code snippet, train test split is performed with 25 percent data in test set. With this we get train data and test data. Train data consists of x underscore train with features, and y underscore train with output. Test data consists of y underscore train with features, and y underscore test with output.

1.19 Code



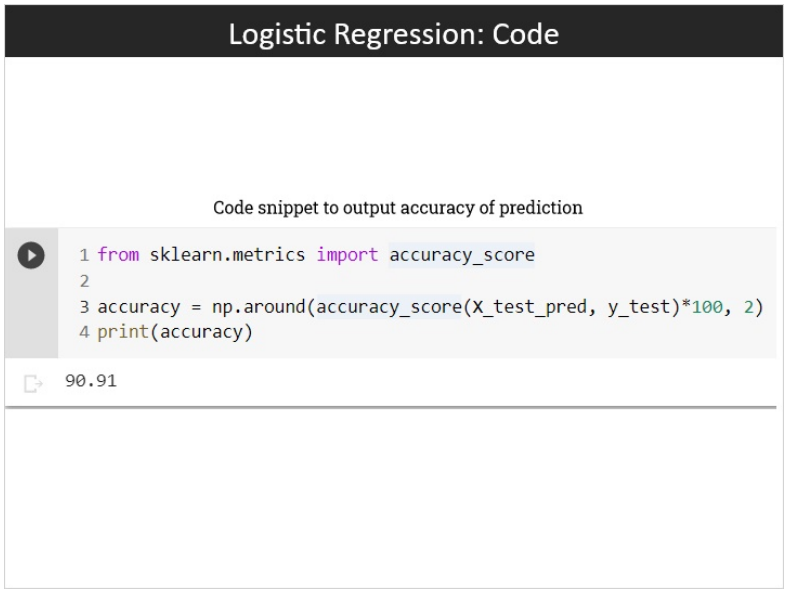
Notes:

Let us discuss the first code snippet. In line number 1, required module named Logistic Regression is imported from s k learn dot linear underscore model. Line number two is blank. In line number 3 model is defined with random underscore state as 0. The model is named classifier. Random state is assigned a value so that internal parameters inside the logistic regression module of s k learn is fixed, and do not change whenever the code is run again. In line number 4, the classifier is fit on train data that is x underscore train, and y underscore train.

In second code snippet, model named classifier is used to make predictions on test data. These predictions are saved as x underscore test underscore pred.

Third code snippet is used to visualize the predictions.

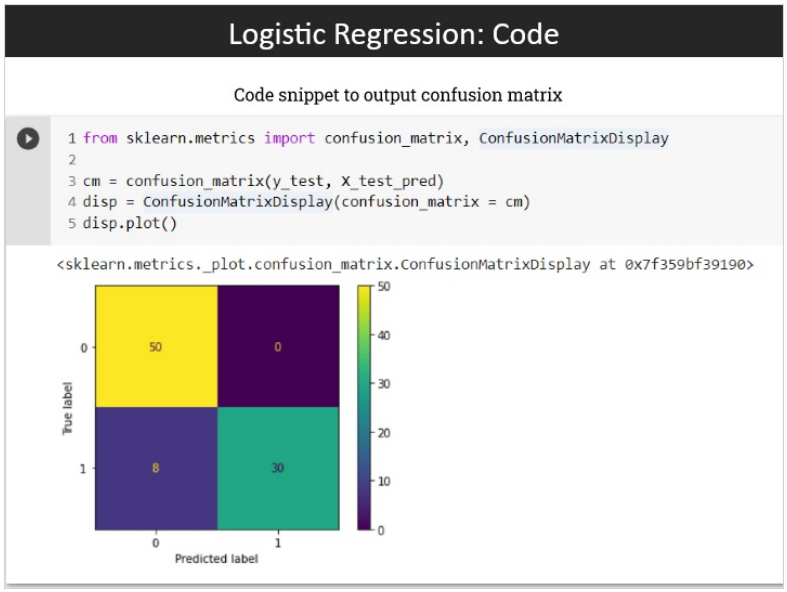
1.20 Code



Notes:

The code snippet on this slide is used to measure accuracy of predictions made on test data. To output accuracy, a module named accuracy underscore score is imported from s k learn dot metrics. Then accuracy is calculated and saved as variable named accuracy. N P dot around is used to round the accuracy value to 2 decimal places. Finally accuracy value is printed which in this case is 90.91.

1.21 Code



Notes:

The code snippet on this slide is used to output confusion matrix. First of all confusion underscore matrix and confusion matrix display modules are imported from s k learn dot metrics. Then confusion matrix is defined using actual output values or y underscore test and predicted values or x underscore test underscore pred. This confusion matrix is saved in a variable named c m. Now an image of confusion matrix is created and saved as disp. Finally disp is rendered to display confusion matrix. Learners are advised to understand the different quadrants of this confusion matrix with the help of theory given earlier. Please note that axes of true label and predicted label may be interchanged in different representations of confusion matrix.

1.22 End

Notes:

Congratulations, You have finished the chapter on Logistic Regression.