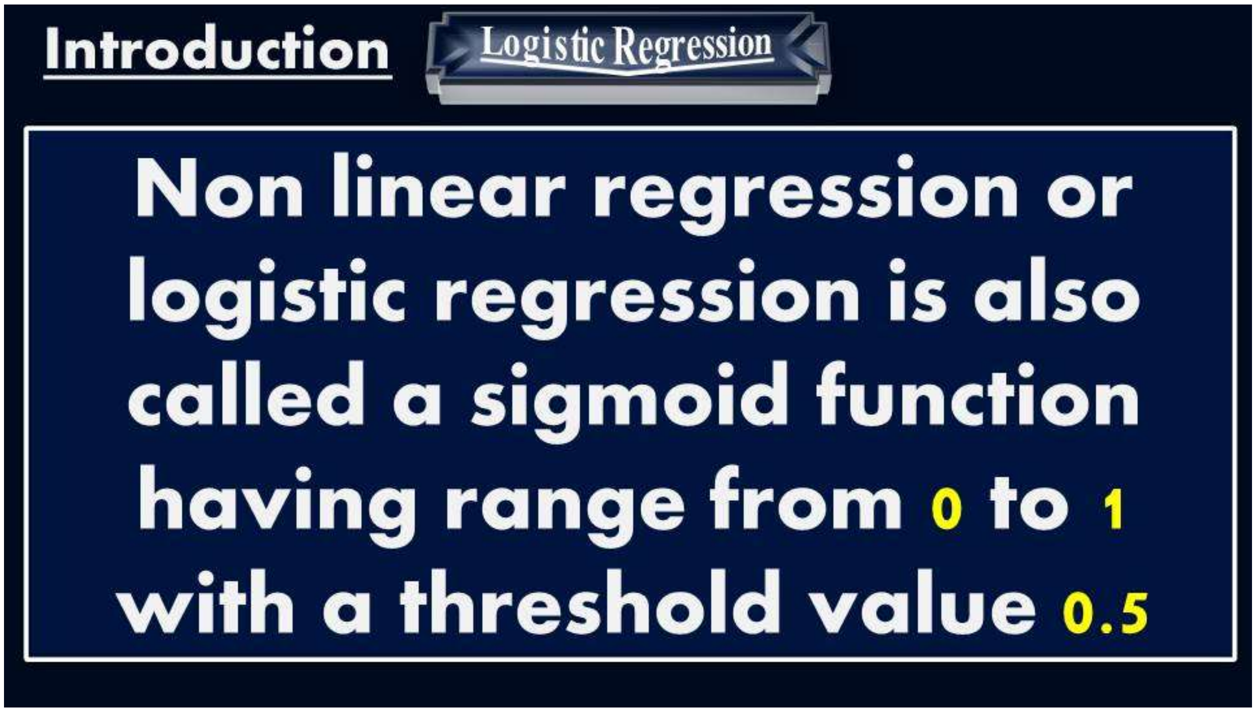
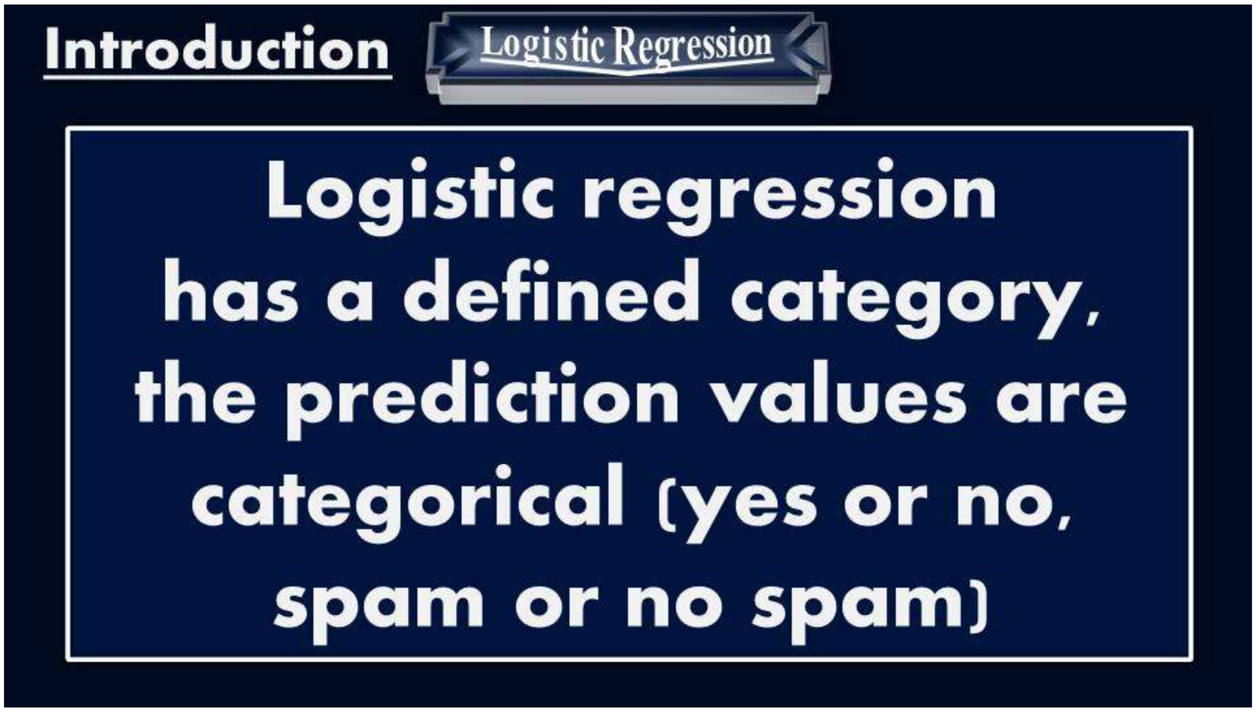


Introduction

Logistic Regression

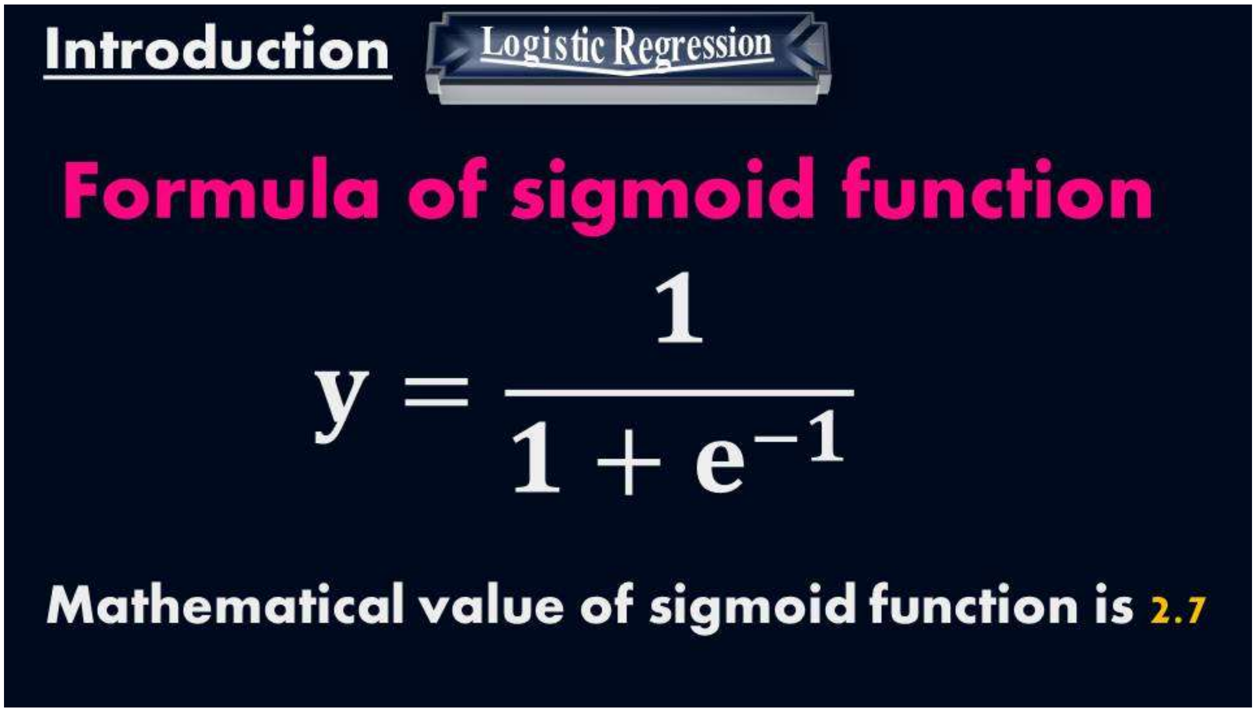
Non linear regression or logistic regression is also called a sigmoid function having range from 0 to 1 with a threshold value 0.5



Introduction

Logistic Regression

Logistic regression has a defined category, the prediction values are categorical (yes or no, spam or no spam)



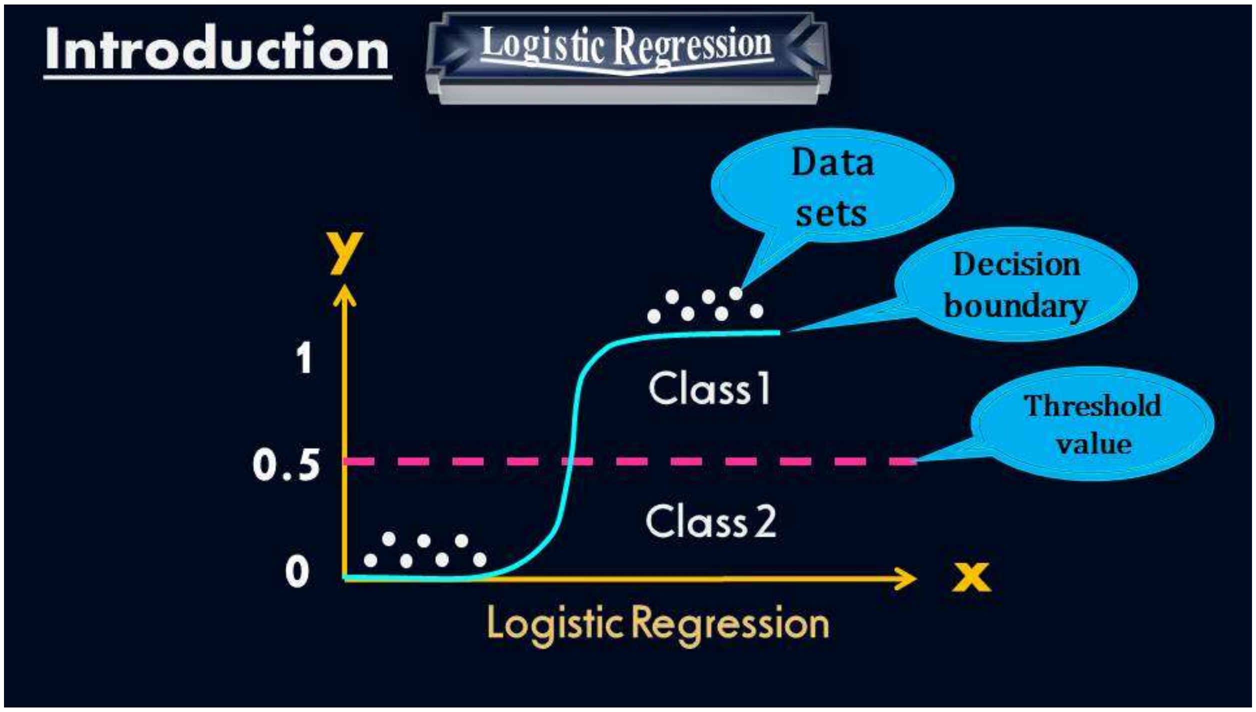
Introduction

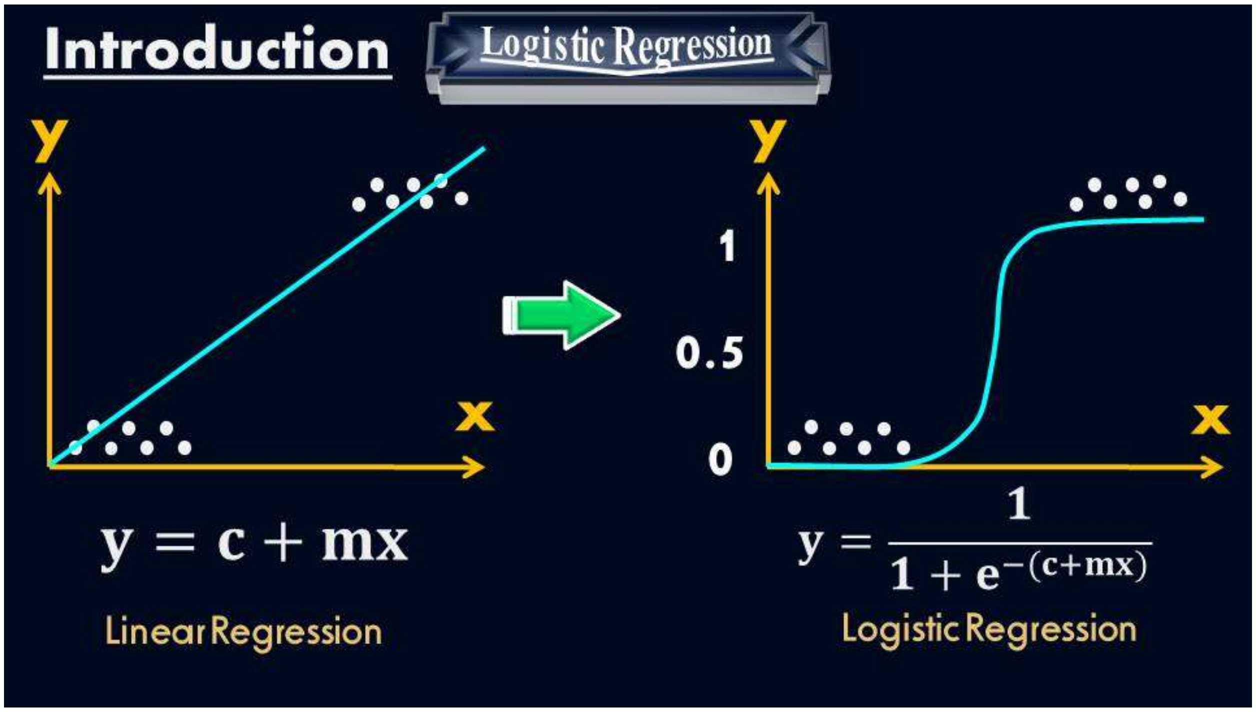
Logistic Regression

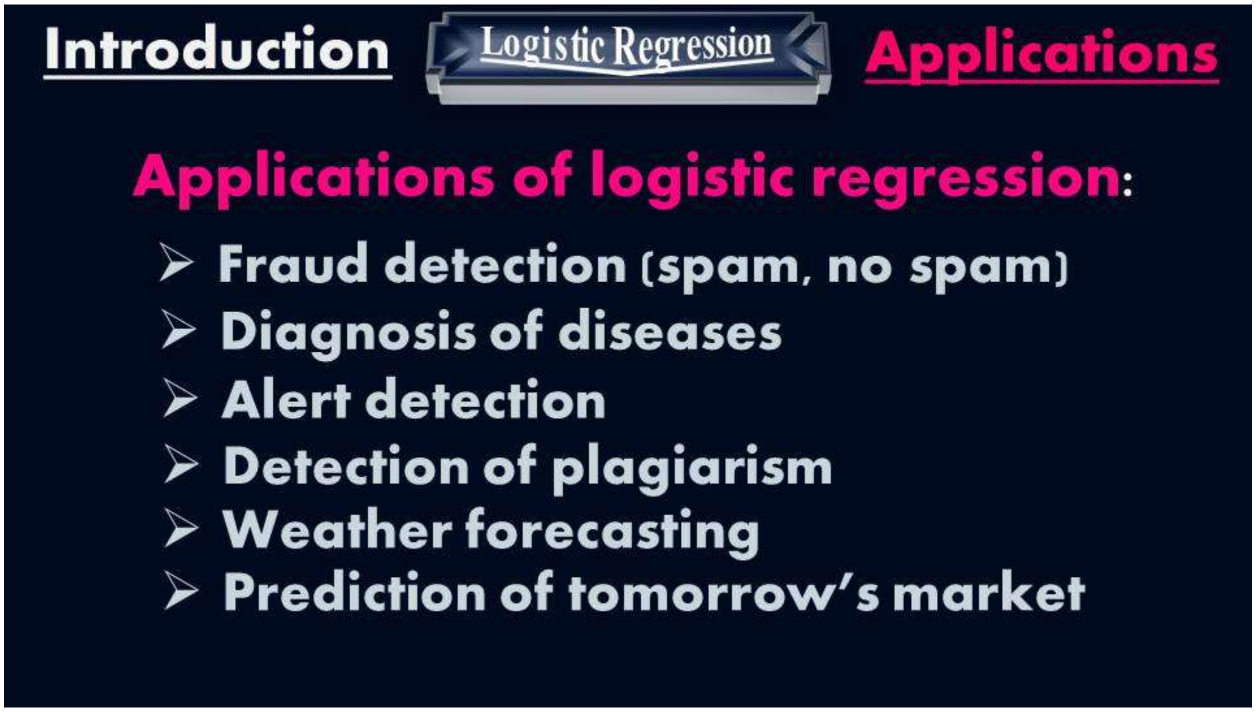
Formula of sigmoid function

$$y = \frac{1}{1 + e^{-1}}$$

Mathematical value of sigmoid function is 2.7





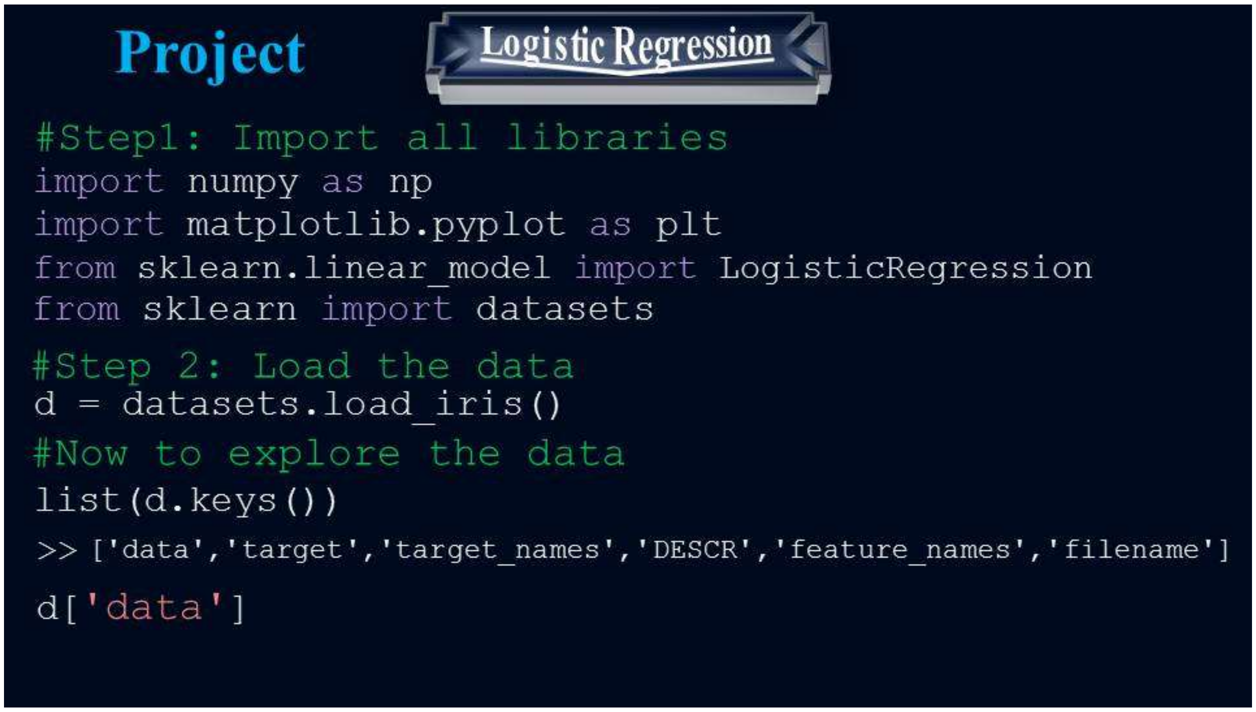

Introduction

Logistic Regression

Applications

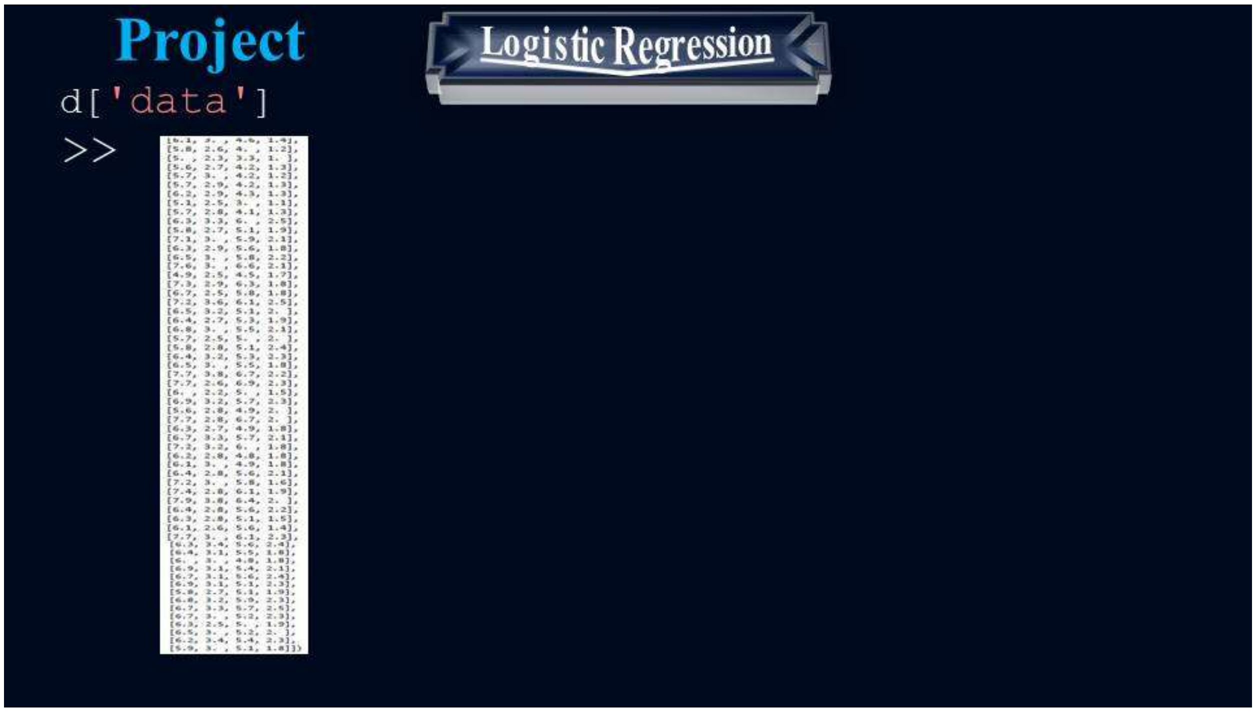
Applications of logistic regression:

- Fraud detection (spam, no spam)
- Diagnosis of diseases
- Alert detection
- Detection of plagiarism
- Weather forecasting
- Prediction of tomorrow's market



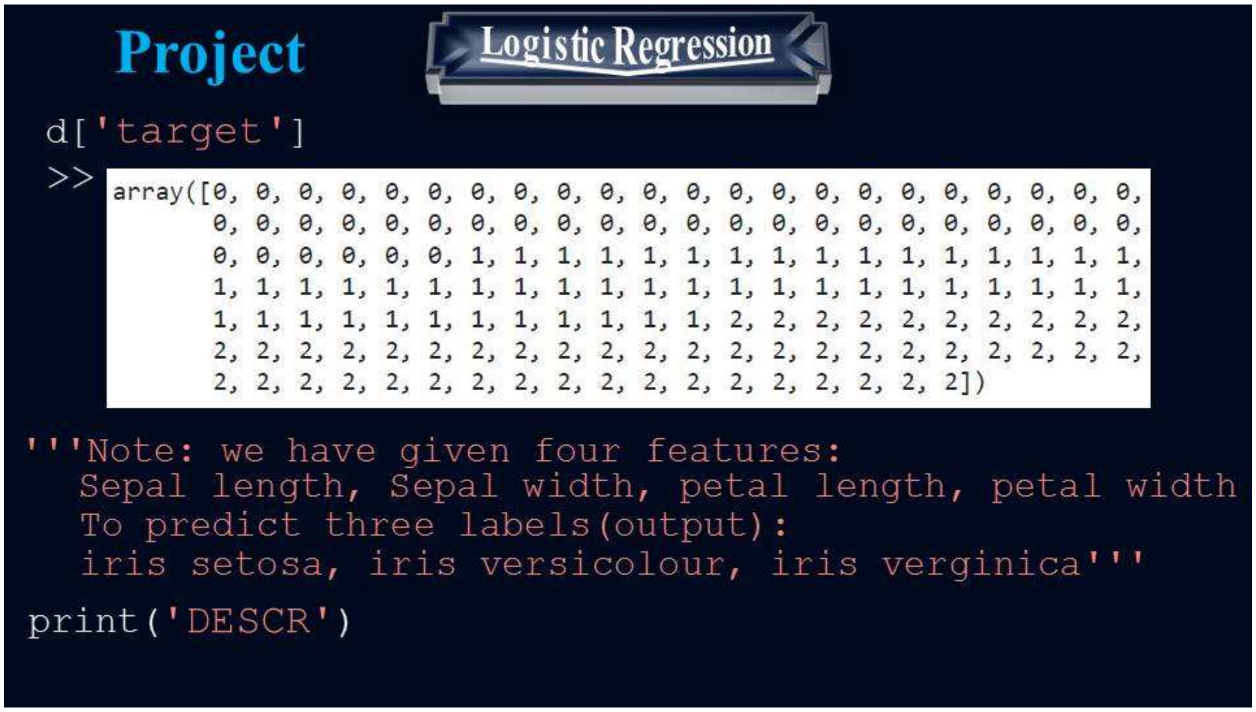
Project Logistic Regression

```
#Step1: Import all libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import datasets

#Step 2: Load the data
d = datasets.load_iris()

#Now to explore the data
list(d.keys())
>> ['data','target','target_names','DESCR','feature_names','filename']

d['data']
```



Project Logistic Regression

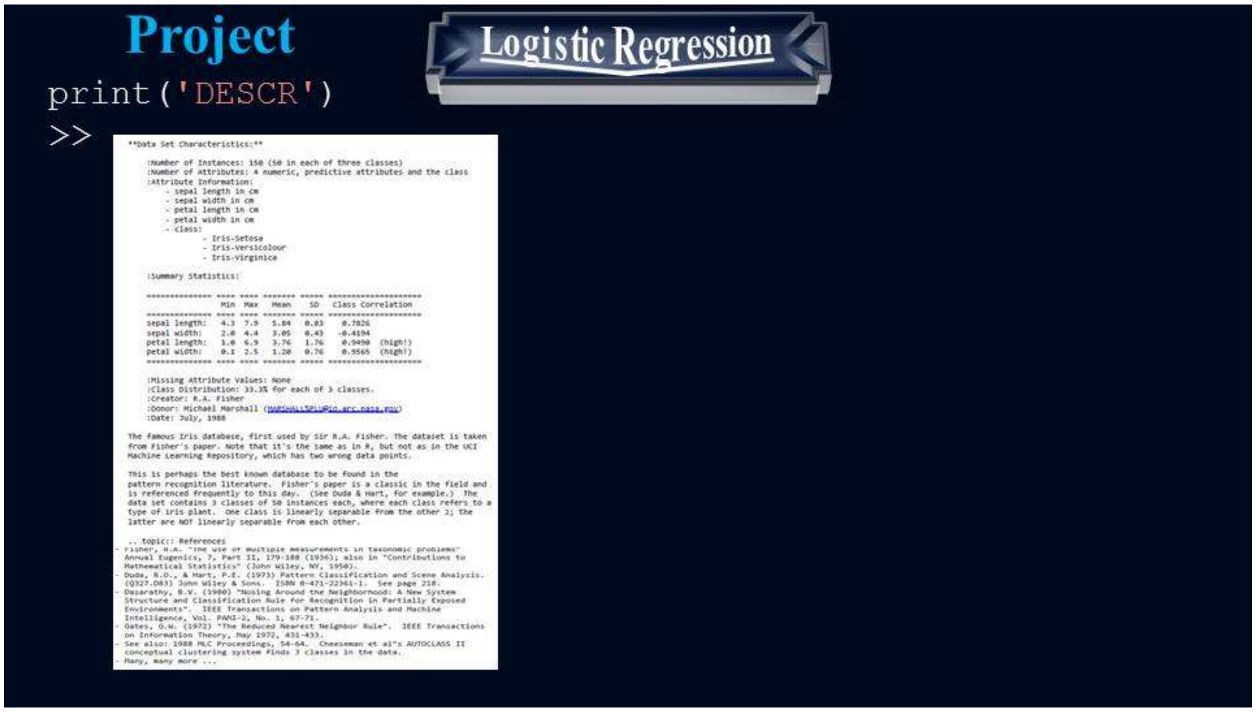
```
d['data']
>>
       [6.1, 3. , 4.6, 1.4],
       [5.8, 2.6, 4. , 1.2],
       [5. , 2.3, 3.3, 1. ],
       [5.6, 2.7, 4.2, 1.3],
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```



Project

Logistic Regression

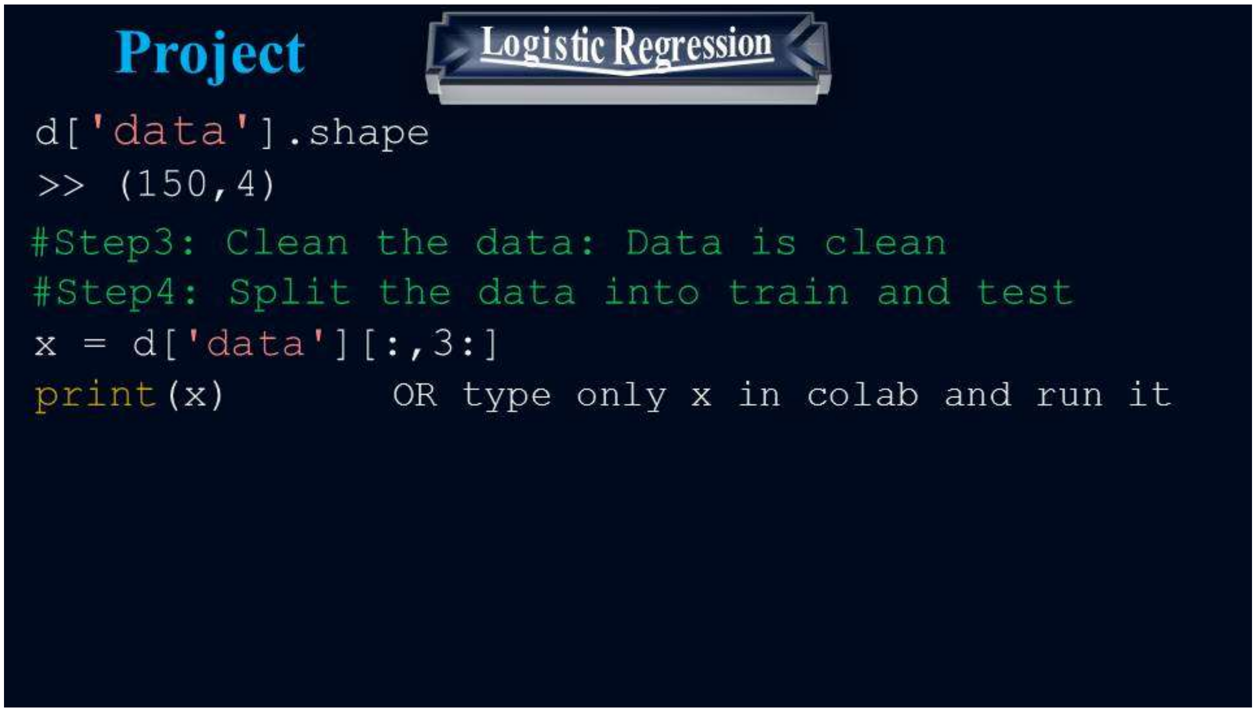
```
d['target']
>>
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

'''Note: we have given four features:
   Sepal length, Sepal width, petal length, petal width
   To predict three labels(output):
   iris setosa, iris versicolour, iris verginica'''

print('DESCR')
```



Project

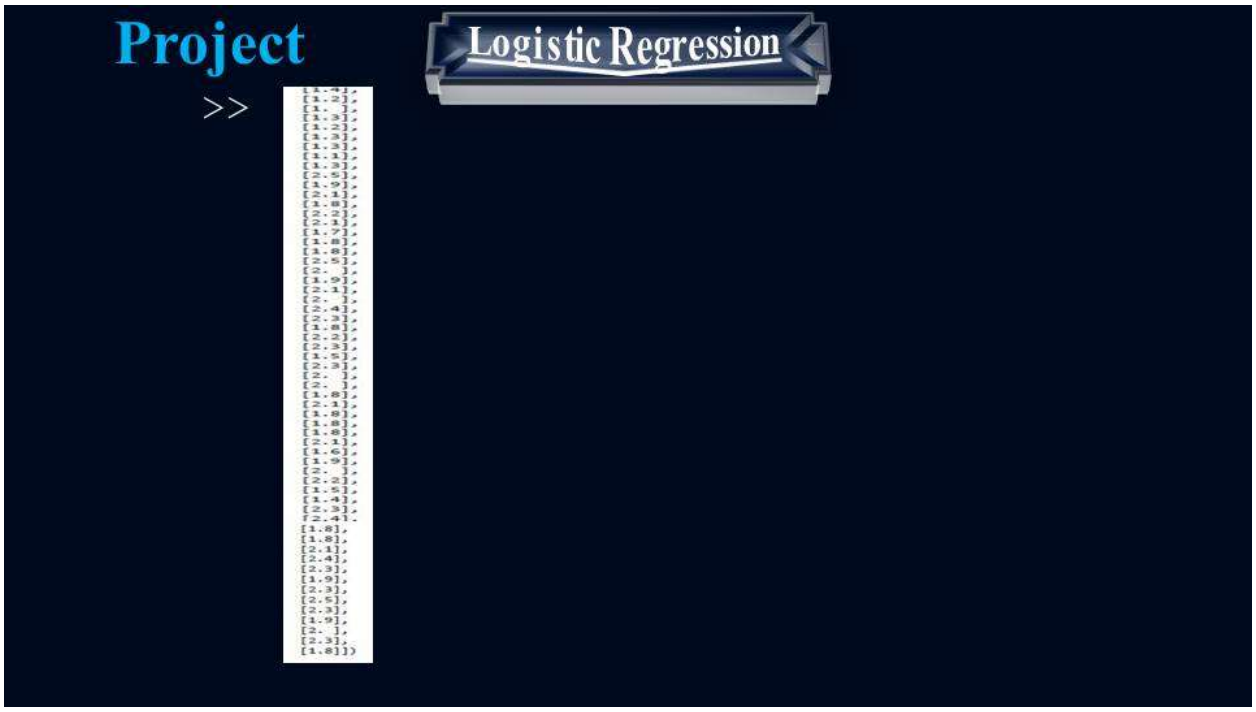
Logistic Regression

```
print('DESCR')
>>
**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References
- Fisher, R.A. "The use of multiple measurements in taxonomic problems"
  Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
  Mathematical Statistics" (John Wiley, NY, 1950).
- Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
  (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
- Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
  Structure and Classification Rule for Recognition in Partially Exposed
  Environments".  IEEE Transactions on Pattern Analysis and Machine
  Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
  on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
  conceptual clustering system finds 3 classes in the data.
- Many, many more ...
```



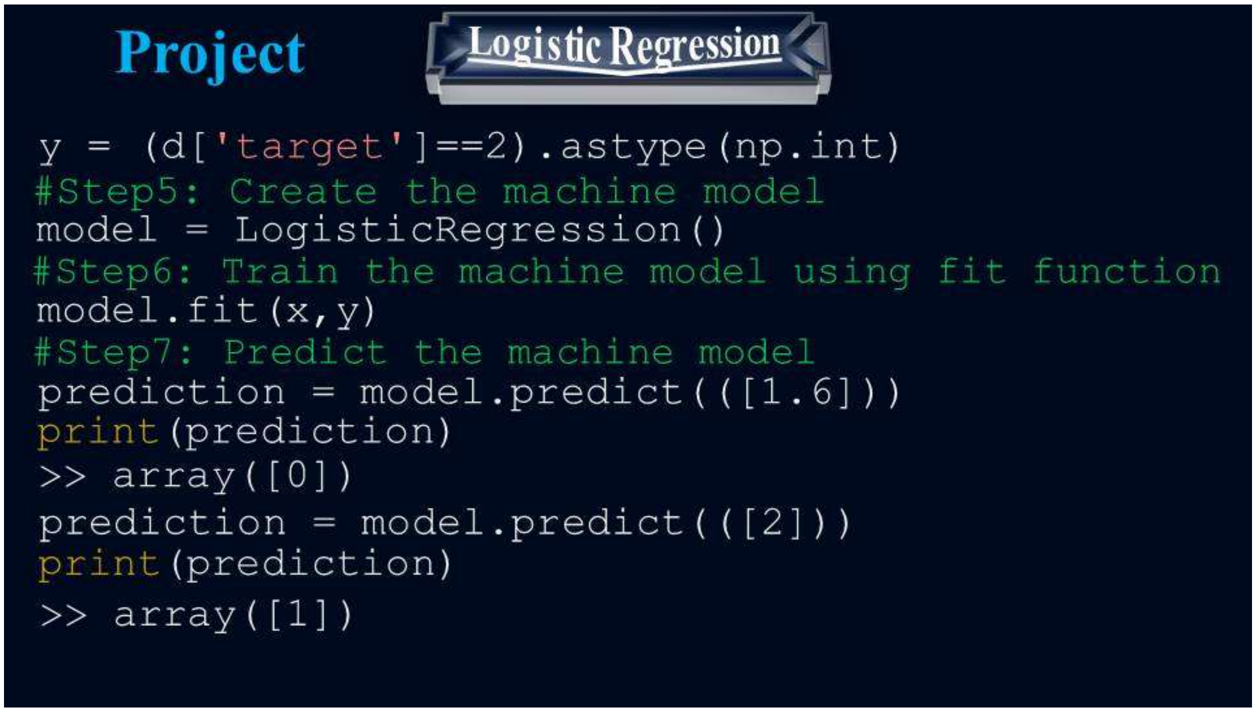
Project Logistic Regression

```
d['data'].shape
>> (150,4)
#Step3: Clean the data: Data is clean
#Step4: Split the data into train and test
x = d['data'][:,3:]
print(x)        OR type only x in colab and run it
```

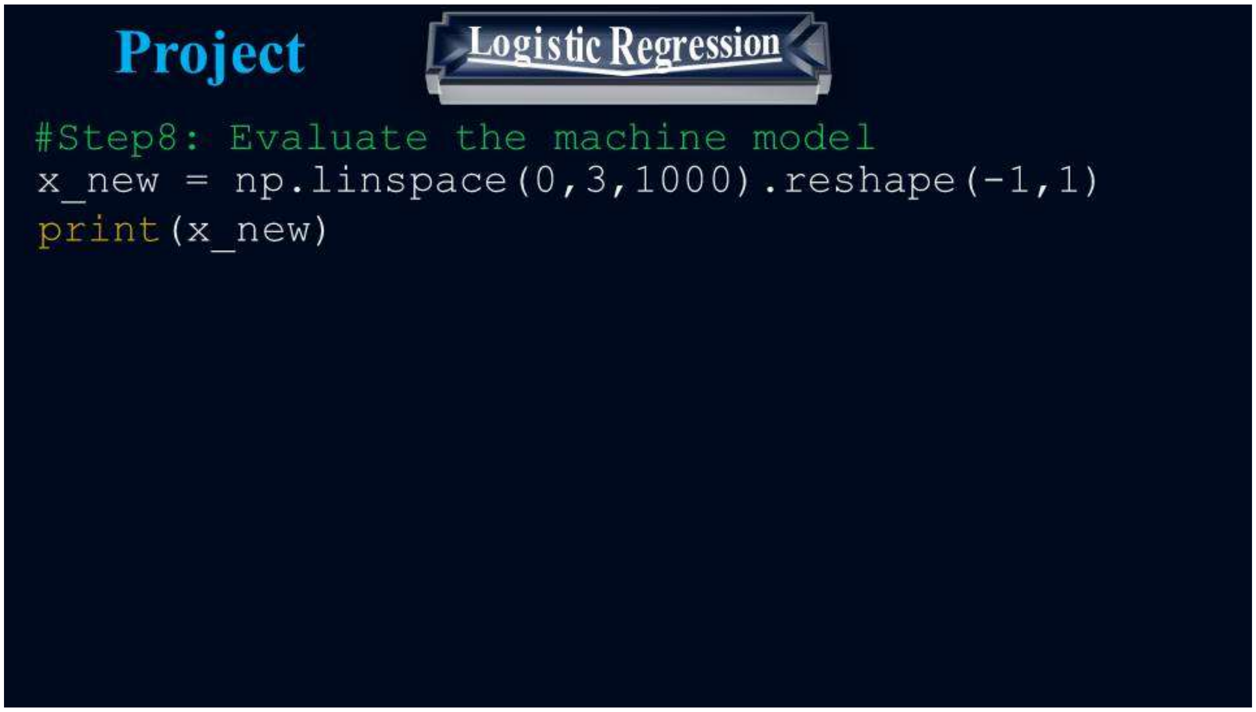


Project Logistic Regression

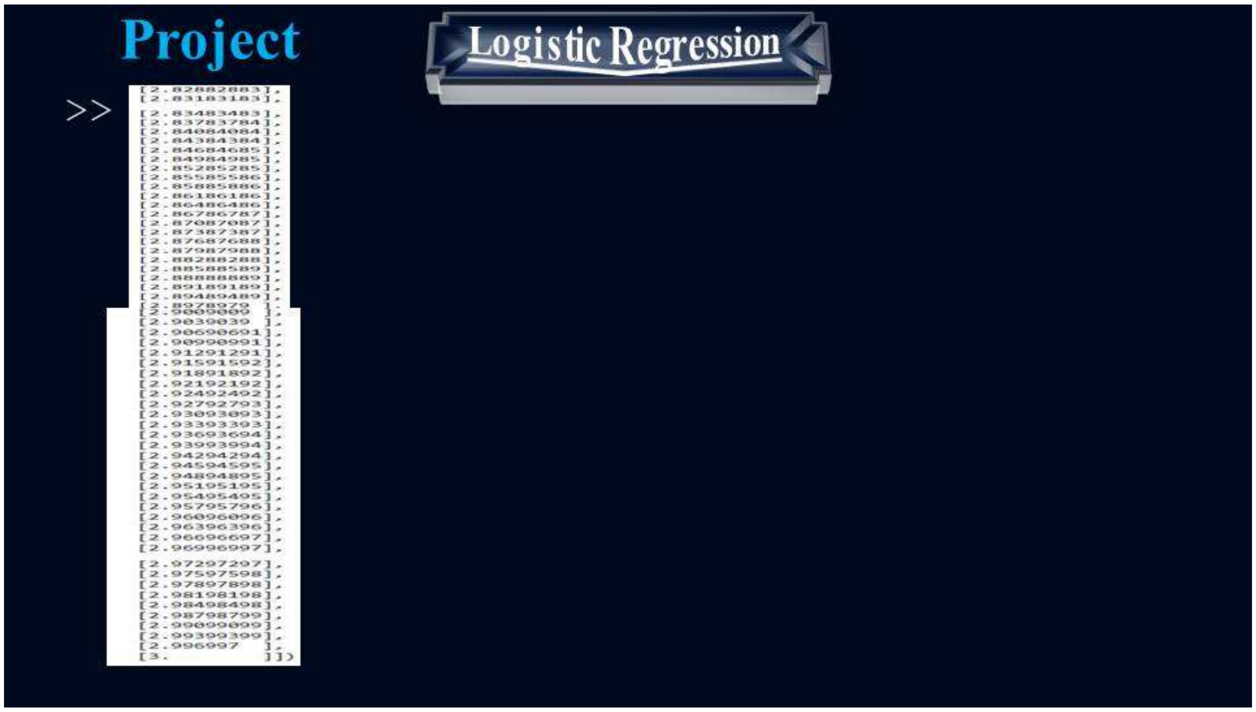
```
>>   [1.4],
     [1.2],
     [1. ],
     [1.3],
     [1.2],
     [1.3],
     [1.3],
     [1.1],
     [1.3],
     [2.5],
     [1.9],
     [2.1],
     [1.8],
     [2.2],
     [2.1],
     [1.7],
     [1.8],
     [1.8],
     [2.5],
     [2. ],
     [1.9],
     [2.1],
     [2. ],
     [2.4],
     [2.3],
     [1.8],
     [2.2],
     [2.3],
     [1.5],
     [2.3],
     [2. ],
     [2. ],
     [1.8],
     [2.1],
     [1.8],
     [1.8],
     [1.8],
     [2.1],
     [1.6],
     [1.9],
     [2. ],
     [2.2],
     [1.5],
     [1.4],
     [2.3],
     [2.4],
     [1.8],
     [1.8],
     [2.1],
     [2.4],
     [2.3],
     [1.9],
     [2.3],
     [2.5],
     [2.3],
     [1.9],
     [2. ],
     [2.3],
     [1.8]])
```



```
y = (d['target']==2).astype(np.int)
#Step5: Create the machine model
model = LogisticRegression()
#Step6: Train the machine model using fit function
model.fit(x,y)
#Step7: Predict the machine model
prediction = model.predict(([1.6]))
print(prediction)
>> array([0])
prediction = model.predict(([2]))
print(prediction)
>> array([1])
```



```
#Step8: Evaluate the machine model
x_new = np.linspace(0,3,1000).reshape(-1,1)
print(x_new)
```



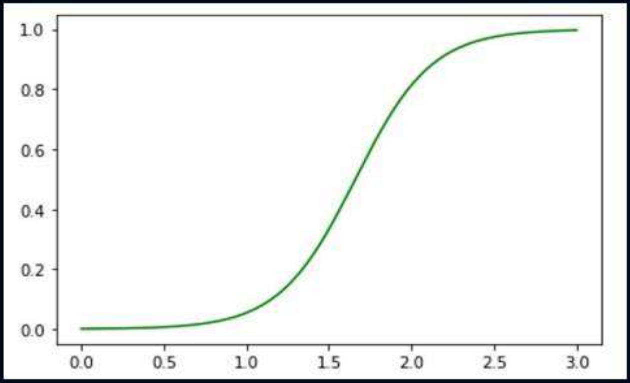
Project Logistic Regression

```
>> [2.82882883],
   [2.83183183],
   [2.83483483],
   [2.83783784],
   [2.84084084],
   [2.84384384],
   [2.84684685],
   [2.84984985],
   [2.85285285],
   [2.85585586],
   [2.85885886],
   [2.86186186],
   [2.86486486],
   [2.86786787],
   [2.87087087],
   [2.87387387],
   [2.87687688],
   [2.87987988],
   [2.88288288],
   [2.88588589],
   [2.88888889],
   [2.89189189],
   [2.89489489],
   [2.8978979 ],
   [2.9009009 ],
   [2.9039039 ],
   [2.90690691],
   [2.90990991],
   [2.91291291],
   [2.91591592],
   [2.91891892],
   [2.92192192],
   [2.92492492],
   [2.92792793],
   [2.93093093],
   [2.93393393],
   [2.93693694],
   [2.93993994],
   [2.94294294],
   [2.94594595],
   [2.94894895],
   [2.95195195],
   [2.95495495],
   [2.95795796],
   [2.96096096],
   [2.96396396],
   [2.96696697],
   [2.96996997],
   [2.97297297],
   [2.97597598],
   [2.97897898],
   [2.98198198],
   [2.98498498],
   [2.98798799],
   [2.99099099],
   [2.99399399],
   [2.996997  ],
   [3.        ]])
```

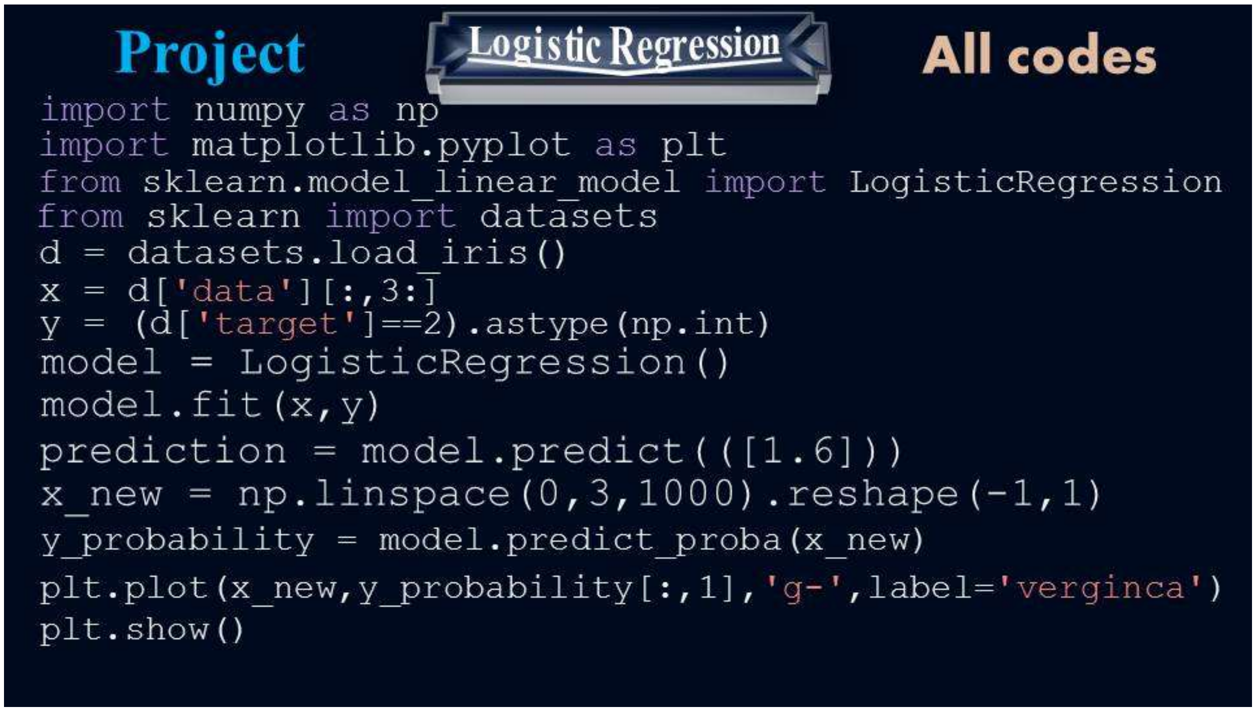
Project Logistic Regression

```
y_probability = model.predict_proba(x_new)
plt.plot(x_new,y_probability[:,1],'g-',label='verginca')
plt.show()
```

>>



COMPLETE CODES ON ONE PAGE



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_linear_model import LogisticRegression
from sklearn import datasets
d = datasets.load_iris()
x = d['data'][:,3:]
y = (d['target']==2).astype(np.int)
model = LogisticRegression()
model.fit(x,y)
prediction = model.predict(([1.6]))
x_new = np.linspace(0,3,1000).reshape(-1,1)
y_probability = model.predict_proba(x_new)
plt.plot(x_new,y_probability[:,1],'g-',label='verginca')
plt.show()
```