```
# R L B P
# What is Logistic Regression?

# A regression algorithm which does classification
# Calculates probability of belonging to a particular class
#  p > 50 % -> 1
#  p < 50 % -> 0
```

```
# Train a logistic regression classfier
# to predict whether a flower is iris virginica or not
```

```
from sklearn import datasets
import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
```

```
iris = datasets.load_iris()
print(list(iris.keys()))
print(iris['data'])
print(iris['target'])
print(iris['DESCR'])
```

```
: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.

.. dropdown:: 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 ...
```

```
print(iris['data'].shape)
```

```
(150, 4)
```

```
X = iris["data"][:, 3:]
print(iris["data"])
print(X)
```

```
 [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 = (iris["target"]==2)
print(y)
```

```
[False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
```

```
False False False False  True  True  True  True  True  True  True  True
 True  True  True  True  True  True  True  True  True  True  True  True
 True  True  True  True  True  True  True  True  True  True  True  True
 True  True  True  True  True  True  True  True  True  True  True  True
 True  True  True  True  True  True]
```

```
# Train a logistic regression classifier
clf = LogisticRegression()
clf.fit(X,y)
```

```
▾ LogisticRegression ⓘ ?
LogisticRegression()
```

```
example =clf.predict([[2.6]])
print(example)
```

```
[ True]
```

```
# Using matplotlib to plot the visualization
X_new = np.linspace(0,3,1000).reshape(1,-1)

print(X_new)
```

```
 2.45045045 2.45345345 2.45645646 2.45945946 2.46246246 2.46546547
 2.46846847 2.47147147 2.47447447 2.47747748 2.48048048 2.48348348
 2.48648649 2.48948949 2.49249249 2.4954955  2.4984985  2.5015015
 2.5045045  2.50750751 2.51051051 2.51351351 2.51651652 2.51951952
 2.52252252 2.52552553 2.52852853 2.53153153 2.53453453 2.53753754
 2.54054054 2.54354354 2.54654655 2.54954955 2.55255255 2.55555556
 2.55855856 2.56156156 2.56456456 2.56756757 2.57057057 2.57357357
 2.57657658 2.57957958 2.58258258 2.58558559 2.58858859 2.59159159
 2.59459459 2.5975976  2.6006006  2.6036036  2.60660661 2.60960961
 2.61261261 2.61561562 2.61861862 2.62162162 2.62462462 2.62762763
 2.63063063 2.63363363 2.63663664 2.63963964 2.64264264 2.64564565
 2.64864865 2.65165165 2.65465465 2.65765766 2.66066066 2.66366366
 2.66666667 2.66966967 2.67267267 2.67567568 2.67867868 2.68168168
 2.68468468 2.68768769 2.69069069 2.69369369 2.6966967  2.6996997
 2.7027027  2.70570571 2.70870871 2.71171171 2.71471471 2.71771772
 2.72072072 2.72372372 2.72672673 2.72972973 2.73273273 2.73573574
 2.73873874 2.74174174 2.74474474 2.74774775 2.75075075 2.75375375
 2.75675676 2.75975976 2.76276276 2.76576577 2.76876877 2.77177177
 2.77477477 2.77777778 2.78078078 2.78378378 2.78678679 2.78978979
 2.79279279 2.7957958  2.7987988  2.8018018  2.8048048  2.80780781
 2.81081081 2.81381381 2.81681682 2.81981982 2.82282282 2.82582583
 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.        ]]
```

```
X_new = np.linspace(0,3,1000).reshape(-1,1)
y_prob = clf.predict_proba(X_new)
# print(y_proba)
plt.plot(X_new, y_prob[:,1], "g-",label="virginica")
plt.show()
```


```
print(y_prob)
```

```
[[9.99249051e-01 7.50949397e-04]
 [9.99239224e-01 7.60776030e-04]
 [9.99229269e-01 7.70731151e-04]
 ...
 [3.08499021e-03 9.96915010e-01]
 [3.04523414e-03 9.96954766e-01]
 [3.00598887e-03 9.96994011e-01]]
```

```
# clf = LogisticRegression()
# clf.fit(X,y)
```

```
# Using matplotlib to plot the visualization
# X_new = np.linspace(0,3,1000).reshape(-1.1)
# print(X_new)
```

```
Start coding or generate with AI.
```