

# Logistic Regression For Multinomial Classification



Problem:

You have data of iris flowers dataset.

It has 3 different types of species:

Setosa , Versicolor and Virginica

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
7.6	3	6.6	2.1	virginica
4.9	2.5	4.5	1.7	virginica
7.3	2.9	6.3	1.8	virginica
6.7	2.5	5.8	1.8	virginica

You are given a flower of following dimensions:

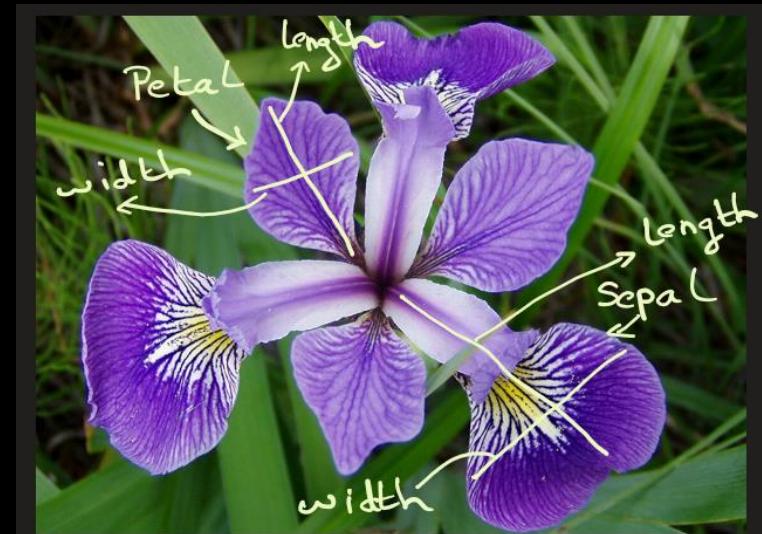
sepal\_length = 6.1

sepal\_width = 2.9

petal\_length = 4.7

petal\_width = 1.4

How would you predict to which category does the above data belongs?





# Logistic Regression For Multinomial Classification



Sometime we have to predict whether a data point belongs to certain category. Category can be

**1. Binomial:** There can be only two possible types of categories

Example: Number can **0 or 1**;

Student either **Pass or Fail**;

email is **spam or not spam**;

Patient has **Cancer or no-cancer**

**2. Multinomial:** There can be 3 or more possible types of categories

Example: Image is either **cat, dogs, or sheep**;

Iris flower can be either **setosa, versicolor or virginica**

We need a model that can classify data points in classes/categories:

This is where **Logistic Regression** come.

**Logistic Regression** is a model that predicts **probability** ( a number between 0 and 1) and then applies a **threshold** (usually 0.5) to determine which class the data belongs to.



# Logistic Regression For Multinomial Classification



Sigmoid function is used in **binary classification** — it outputs a single probability between 0 and 1. But what if we have **3 classes or more**? This is multinomial problem.

We need a function that

- Outputs **one probability per class**, and ensures all probabilities add up to **1**.

That's exactly what **Soft-max function** does.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

The Soft-max function is a mathematical function that converts a **vector of real numbers**, say  $(z_1, z_2, z_3)$ , into a probability distribution.

- Each element in the output is between 0 and 1

$$(z_1, z_2, z_3) \rightarrow \left( \frac{e^{z_1}}{\sum e^{z_i}}, \frac{e^{z_2}}{\sum e^{z_i}}, \frac{e^{z_3}}{\sum e^{z_i}} \right)$$

- The sum of all elements equals 1.

$$\frac{e^{z_1}}{\sum e^{z_i}} + \frac{e^{z_2}}{\sum e^{z_i}} + \frac{e^{z_3}}{\sum e^{z_i}} = 1$$

⋮



# Logistic Regression For Multinomial Classification



Few important points to note about **Soft-max function**:

- $z_i$  : It is a raw score (a.k.a. “logit”) for class  $i$ . It can be positive or negative.
- $e^{z_i}$  : exponentiates the score to make it positive
- The denominator makes sure all probabilities sum to 1:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

$$\sum_{i=1}^K \text{softmax}(z_i) = 1$$

# Logistic Regression For Multinomial Classification

Step by step intuition for 3 class problem:

**Step1:** For a given input  $(x) \rightarrow (z_1, z_2, z_3)$

Each class has its own linear function:

$$z_k = w_k * x + b_k$$

So,

$$z_1 = w_1 * x + b_1$$

$$z_2 = w_2 * x + b_2$$

$$z_3 = w_3 * x + b_3$$

The  $w_k$  and  $b_k$  are calculated during training from training data.

**Step2:** Apply soft-max function:  $(z_1, z_2, z_3) \rightarrow (\frac{e^{z_1}}{\sum e^{z_i}}, \frac{e^{z_2}}{\sum e^{z_i}}, \frac{e^{z_3}}{\sum e^{z_i}})$ .

Based on the probability make a prediction.

# Logistic Regression For Multinomial Classification

$$x \rightarrow (z_1, z_2, z_3) \rightarrow \left( \frac{e^{z_1}}{\sum e^{z_i}}, \frac{e^{z_2}}{\sum e^{z_i}}, \frac{e^{z_3}}{\sum e^{z_i}} \right)$$

# Logistic Regression For Multinomial Classification

Example: Build a **classifier** that recognizes **three fruits** based on weight. The training data is given below. If you have a fruit of weight 150, this classifier should be able to determine the type of the fruit ?

Weight	Fruit
120	Apple
135	Apple
150	Apple
160	Banana
175	Banana
190	Banana
55	Grape
65	Grape
70	Grape

## Step1:

Let's say we have calculated from training data following parameters:

$$(w_1, w_2, w_3) = (-0.1309, 0.4707, -0.3398) \text{ and } (b_1, b_2, b_3) = (24.42, -68.83, 44.41).$$

Then for a given data input  $x = 150$ , the linear scores for 3 classes would be

$$z_k = w_k * x + b_k$$

Plugging in the numbers,

$$(z_1, z_2, z_3) = (4.78, 1.77, -6.56)$$

Where subscript 1,2,3 means apple, banana and grape.

# Logistic Regression For Multinomial Classification



Step 2: Apply the Soft-max Function

Soft-max converts these raw scores into probabilities:

$$(z_1, z_2, z_3) \rightarrow \left( \frac{e^{z_1}}{\sum e^{z_i}}, \frac{e^{z_2}}{\sum e^{z_i}}, \frac{e^{z_3}}{\sum e^{z_i}} \right)$$

<u>Fruit</u>	<u><math>z_i</math></u>	<u><math>e^{z_i}</math></u>	<u>Probability</u>
Apple →	4.78	119.1	$\frac{e^{z_i}}{\sum e^{z_i}} = \frac{119.1}{124.97} = 0.953$
Banana →	1.77	5.87	$\frac{e^{z_i}}{\sum e^{z_i}} = \frac{5.87}{124.97} = 0.047$
Grape →	-6.56	.0014	$\frac{e^{z_i}}{\sum e^{z_i}} = \frac{0.0014}{124.97} = 0.00001$

$$\begin{aligned}\text{Sum, } \sum e^{z_i} &= 119.1 + 5.87 + .0014 \\ &= 124.97\end{aligned}$$

It is **95.3%** chance that it is apple.  
The image is classified as an apple.



## Applying python code



```
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression

x = np.array([120, 135, 150, 160, 175, 190, 55, 65, 70]).reshape(-1, 1)
y_fruits = np.array([
    "Apple", "Apple", "Apple",
    "Banana", "Banana", "Banana",
    "Grape", "Grape", "Grape"
])

# Encode fruit labels as numbers
le = LabelEncoder()
y = le.fit_transform(y_fruits) # converts to 0,1,2

print("Label Mapping:")
for fruit, label in zip(le.classes_, range(len(le.classes_))):
    print(fruit, "->", label, end="; ")

# Train multinomial logistic regression
model = LogisticRegression(multi_class="multinomial", solver="lbfgs")
model.fit(x, y)

# Predict for a new fruit
weight_new = np.array([[150]]) # test weight
probs = model.predict_proba(weight_new)
probs = np.round(probs, 6)

print("Predicted probabilities for weight 150g:")
print(f"Apple : {probs[0][0]}; Banana: {probs[0][1]}; Grape : {probs[0][2]}")
print("\nPredicted Class:", le.inverse_transform(model.predict(weight_new))[0])

Label Mapping:
Apple -> 0; Banana -> 1; Grape -> 2; Predicted probabilities for weight 150g:
Apple : 0.952939; Banana: 0.047049; Grape : 1.1e-05

Predicted Class: Apple
```

# Logistic Regression For Multinomial Classification

So far, we assumed that data has only one feature. Now we can apply the same logic to data with many features.

When you have **one feature**, the model equation is:

$$z = w_1x_1 + b$$

When you have **more than one feature**, say:

- $x_1$  = weight
- $x_2$  = color score
- $x_3$  = sweetness

Then the logistic regression model becomes:

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

**General formula:**

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Or in vector form:

$$z = \mathbf{w}^T \mathbf{x} + b$$



# Logistic Regression For Multinomial Classification



When you have **more than two classes**, like:

- Apple
- Banana
- Grape

You compute **one z-score per class**:

$$z_k = w_{k1}x_1 + w_{k2}x_2 + \cdots + w_{kn}x_n + b_k$$

For example:

**Apple:**

$$z_{apple} = w_{a1}x_1 + w_{a2}x_2 + w_{a3}x_3 + b_a$$

**Banana:**

$$z_{banana} = w_{b1}x_1 + w_{b2}x_2 + w_{b3}x_3 + b_b$$

**Grape:**

$$z_{grape} = w_{g1}x_1 + w_{g2}x_2 + w_{g3}x_3 + b_g$$

Then apply soft-max function:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$



# Logistic Regression For Multinomial Classification



Suppose we feed the following training data to the model:

<u>Weight</u>	<u>Sweetness</u>	<u>ColorIntensity</u>	<u>Fruit</u>
120	8.2	0.4	Apple
155	5.5	0.8	Banana
65	7.0	0.2	Grape

Let's assume the model learned these weights:

**For Apple:**

$$z_{apple} = (0.01)x_1 + (0.5)x_2 + (1.2)x_3 - 5$$

**For Banana:**

$$z_{banana} = (0.02)x_1 + (-0.1)x_2 + (0.3)x_3 - 2$$

**For Grape:**

$$z_{grape} = (-0.03)x_1 + (0.2)x_2 + (0.8)x_3 + 1$$



# Logistic Regression For Multinomial Classification



Step1:

Now for a new fruit:

$$x = (130, 6.0, 0.6)$$

Compute:

**Apple**

$$z_a = 0.01(130) + 0.5(6) + 1.2(0.6) - 5$$

**Banana**

$$z_b = 0.02(130) - 0.1(6) + 0.3(0.6) - 2$$

**Grape**

$$z_g = -0.03(130) + 0.2(6) + 0.8(0.6) + 1$$

Simplifying above, we get

$$x = (130, 6, 0.6) \rightarrow (z_a, z_b, z_g) = (0.02, 0.18, -1.22)$$



# Logistic Regression For Multinomial Classification



Step2: Then apply softmax to convert z-values to probabilities.

$$(z_a, z_b, z_g) = (0.02, 0.18, -1.22)$$

$$e^{0.02} \approx 1.0202$$

$$e^{0.18} \approx 1.1972$$

$$e^{-1.22} \approx 0.2953$$

The Sum,  $\sum e^{z_i} = 1.0202 + 1.1972 + 0.2953 = 2.5127$

$$(z_1, z_2, z_3) \rightarrow \left( \frac{e^{z_1}}{\sum e^{z_i}}, \frac{e^{z_2}}{\sum e^{z_i}}, \frac{e^{z_3}}{\sum e^{z_i}} \right)$$

$$\begin{aligned}(0.02, 0.18, -1.22) &\rightarrow \left( \frac{1.0202}{2.5127}, \frac{1.1972}{2.5127}, \frac{0.2953}{2.5127} \right) \\ &= (0.406, 0.476, 0.117)\end{aligned}$$

The class that has highest probability, that class gets assigned to this fruit. In this case, it is Banana



## Applying python code to iris dataset.



```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from scipy.special import softmax

# 1. Read the CSV file
df = pd.read_csv("data_iris.csv")

# 2. Split features (X) and target (y)
X = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].values
y = df['species'].values

# 3. Encode target labels (e.g., setosa → 0, versicolor → 1, virginica → 2)
encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)

# 4. Train a multinomial logistic regression model (uses Softmax internally)
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=200)
model.fit(X, y_encoded)

# 5. Take one sample (sepal_length, sepal_width, petal_length, petal_width) = (6.1, 2.9, 4.7, 1.4)
# and get the raw model scores (logits)
sample = np.array([6.1, 2.9, 4.7, 1.4]).reshape(1, -1) # This should be versicolor
z = model.decision_function(sample)
print("Raw model scores (logits):", z)

# 6. Apply softmax manually to convert scores → probabilities
probs = softmax(z)
print("Softmax probabilities:", probs)
print("Predicted class index:", np.argmax(probs))
print("Predicted class label:", encoder.inverse_transform([np.argmax(probs)])[0])
```

Raw model scores (logits): [[-3.27282176 2.27379744 0.99902432]]

Softmax probabilities: [[0.0030393 0.77918334 0.21777736]]

Predicted class index: 1

Predicted class label: versicolor

# EXTRA

Jlds

# Logistic Regression For Multinomial Classification



Jlds

# Logistic Regression For Multinomial Classification





$$\sum_{i=1}^K \text{softmax}(z_i) = 1$$



# Logistic Regression For Multinomial Classification



The general idea of logistic regression:

For example, if you're classifying an image of tumor (x) into one of 3 categories (tumorA, tumorB, tumorC), then

Step1:

Image (x)  $\rightarrow$   $(z_1, z_2, z_3)$ . One value for each class

Step2:

Apply softmax function to this vector:  $(z_1, z_2, z_3) \rightarrow (0.60, 0.15, 0.25)$

Meaning that there's a  
60% chance the image is a tumorA,  
15% chance the image is a tumorB,  
25% chance the image is a tumorC.

So, the image is tumorA

$$x \rightarrow (z_1, z_2, z_3) \rightarrow \left( \frac{e^{z_1}}{\sum e^{z_i}}, \frac{e^{z_2}}{\sum e^{z_i}}, \frac{e^{z_3}}{\sum e^{z_i}} \right)$$