Fondamenti di Analisi dei Dati

from data analysis to predictive techniques

Prof. Antonino Furnari (antonino.furnari@unict.it)
Corso di Studi in Informatica
Dip. di Matematica e Informatica
Università di Catania

Università di Catania

Multiclass Logistic Regression

Extending logistic regression beyond binary classification to handle multiple classes, and exploring the algorithm from a purely predictive machine learning perspective.

The Diabetes Dataset

In many cases, we may need to study the relationship between some dependent variables and an independent one which is **categorical with more than two possible levels**. Let's consider the diabetes dataset:

	relwt	glufast	glutest	instest	sspg	group
0	0.81	80	356	124	55	Normal
1	0.95	97	289	117	76	Normal
2	0.94	105	319	143	105	Normal
3	1.04	90	356	199	108	Normal
4	1.00	90	323	240	143	Normal

relwt

Relative weight (ratio of actual to expected weight)

glufast

Fasting plasma glucose level

glutest

Glucose intolerance (oral glucose tolerance test area)

instest

Insulin response (insulin area)

sspg

Steady state plasma glucose (measure of insulin resistance)

group

Three diagnostic group: `Normal`, `Chemical_Diabetic`, or `Overt_Diabetic`

Multinomial Logistic Regression

The multinomial logistic regression model extends binary logistic regression by modeling the log-odds ratio between each class k and a baseline class (typically class 1):

$$\log\left(\frac{P(Y=k|X=\mathbf{x})}{P(Y=1|X=\mathbf{x})}\right) = \beta_{k0} + \beta_{k1}x_1 + \ldots + \beta_{kn}x_n$$

The model estimates **(n+1) × (K-1)** parameters, where n is the number of predictors and K is the number of classes. Each non-baseline class gets its own set of coefficients.

Through mathematical derivation, we can express the class probabilities as:

Non-baseline classes

$$P(Y=k|X=\mathbf{x}) = \frac{e^{\beta_\mathbf{k}^T\mathbf{x}}}{1+\sum_{l=2}^{K}\mathbf{e}^{\beta_\mathbf{l}^T\mathbf{x}}}$$

Baseline class

$$P(Y=1|X=\mathbf{x}) = \frac{1}{1+\sum_{l=2}^{K}\mathbf{e}^{\beta_l^\mathbf{T}\mathbf{x}}}$$

These expressions compute class probabilities once parameters are estimated and predictors are observed.

Optimisation We won't see the details, but this is optimised by establishing a similar objective (maximum likelihood or cross-entropy).

Multinomial Logistic Regression Example

If we fit this model with statsmodels:

$$\text{group} = \beta_0 + \beta_1 \text{relwt} + \beta_1 \text{glutest}$$

We obtain the following output:

```
Optimization terminated successfully.
         Current function value: 0.100843
         Iterations 14
                          MNLogit Regression Results
==============================================================================
Dep. Variable:                  group   No. Observations:                  145
Model:                        MNLogit   Df Residuals:                      141
Method:                           MLE   Df Model:                            2
Date:                Sat, 15 Nov 2025   Pseudo R-squ.:                  0.9013
Time:                        15:53:12   Log-Likelihood:                -14.622
converged:                       True   LL-Null:                       -148.10
Covariance Type:            nonrobust   LLR p-value:                 1.076e-58
==============================================================================
   group=1       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -90.3738     42.527     -2.125      0.034    -173.726      -7.022
glutest        0.2153      0.101      2.128      0.033       0.017       0.413
------------------------------------------------------------------------------
   group=2       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -109.8266     43.021     -2.553      0.011    -194.147     -25.506
glutest        0.2471      0.102      2.428      0.015       0.048       0.447
==============================================================================

Model Likelihood Ratio Test:
266.9538631863749 1.075734625201075e-58
```

Model Overview

Model Convergence

Optimization terminated successfully after 14 iterations with log-likelihood of -14.622

Likelihood Ratio Test

Statistic: 266.95 with p-value ≈ 0, showing highly significant improvement over null model

Pseudo R^2

This is $R^2 = 1 - \frac{-14.622}{-148.10} = 0.9013$. The multinomial logistic regressor explains about 90% of the uncertainty in the outcomes.

Two Coefficient Sets

Separate coefficients estimated for Chemical_Diabetic and Overt_Diabetic groups versus Normal baseline

Coefficient Interpretation

Chemical Diabetic Intercept: -90.37

When glutest is zero, odds of being Chemical_Diabetic vs Normal are $e^{-90.37} \approx 0$, indicating extremely low probability. The subject is almost certainly Normal at baseline.

Overt Diabetic Intercept: -109.83

When glutest is zero, odds of being Overt_Diabetic vs Normal are $e^{-109.83} \approx 0$, an even lower probability than Chemical_Diabetic.

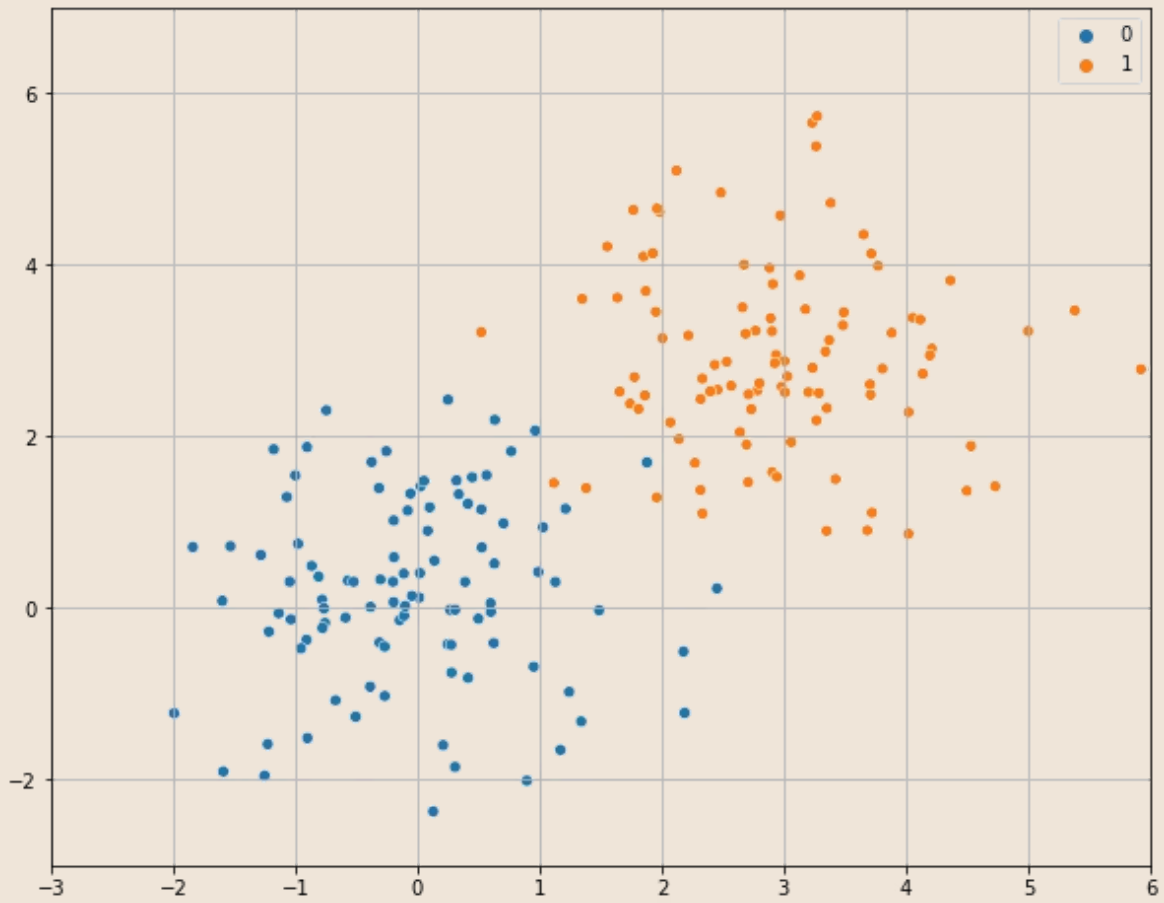
Chemical Diabetic Slope: 0.2153

Each one-unit increase in glutest multiplies the odds by $e^{0.2153} \approx 1.24$, representing a 24% increase in odds of Chemical_Diabetic vs Normal.

Overt Diabetic Slope: 0.2471

Each one-unit increase in glutest multiplies the odds by $e^{0.2471} \approx 1.28$, representing a 28% increase in odds of Overt_Diabetic vs Normal.

Geometrical Interpretation of the Coefficients of a Logistic Regressor



While the logistic function itself is non-linear, **logistic regression is fundamentally a linear classifier**. The decision boundary—where P(y=1|x) = 0.5—forms a hyperplane in feature space.

The figure on the left shows an example. A linear classifier would find a line separating the two sets of points.

We will see that the logistic regressor does just that.

We define the decision boundary as the set of points of maximum uncertainty, i.e., $P(y=1|\mathbf{x}) = 0.5$. Recalling that in the case of two dimensions:

$$P(y=1|\mathbf{x}) = \frac{1}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}$$

For two-dimensional data, this boundary is hence a straight line defined by:

$$P\left(y=1|\mathbf{x}\right) = 0.5 \Leftrightarrow e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)} = 1 \Leftrightarrow 0 = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

This can also be rewritten as follows:

$$x_2 = -\frac{\beta_1}{\beta_2} x_1 - \frac{\beta_0}{\beta_2}$$

The coefficients determine both the orientation (slope) and position (intercept) of this separating line (the decision boundary).

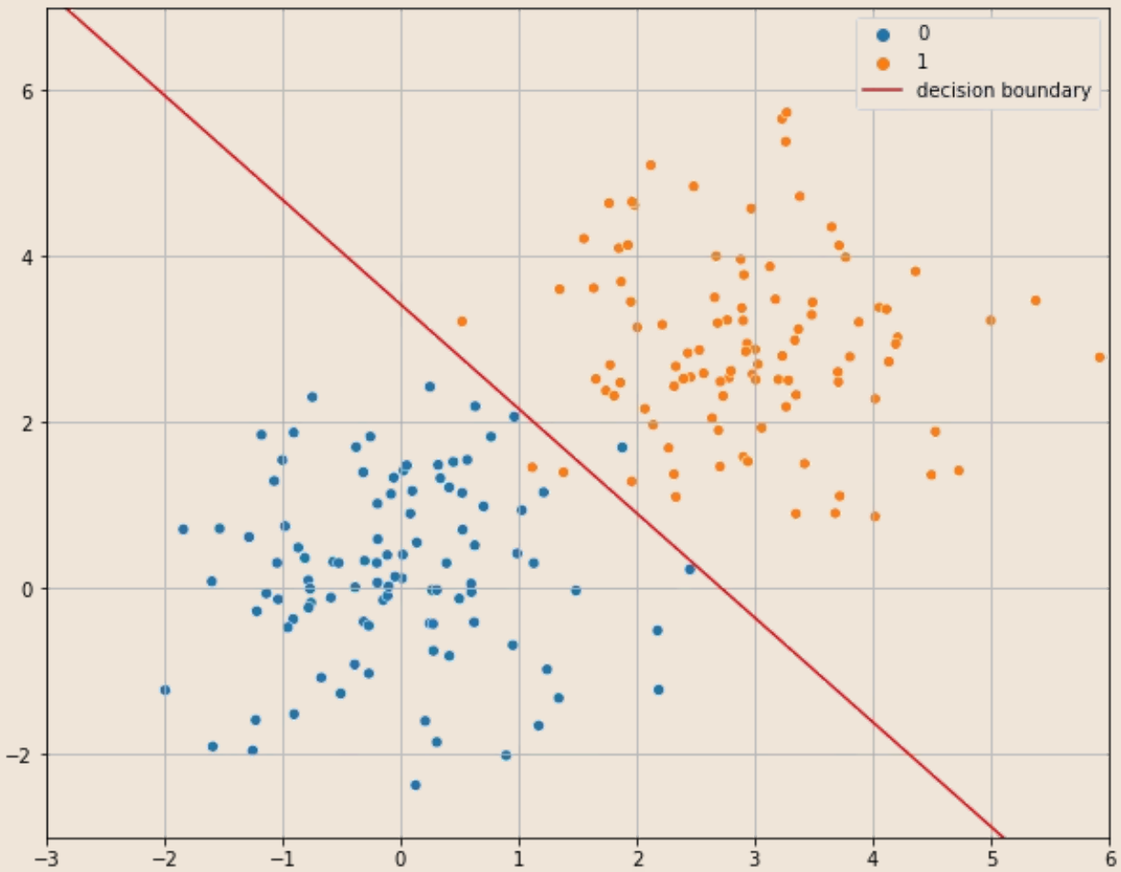
Decision Boundary Example

For example, let's assume we find the following parameters for the logistic regressor $P(y = 1|x_1, x_2) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$:

β_0	β_1	β_2
-3.47	1.17	1.43

These parameters define the following decision boundary:

$$x_2 = -\frac{1.17}{1.43} + \frac{3.47}{1.43} = 2.43x_1 - 0.82$$



The Softmax Regressor for Multiclass Problems

Softmax regression provides an alternative formulation to multinomial logistic regression that avoids selecting a baseline class. Instead of estimating K-1 coefficient sets, it estimates K sets symmetrically.

This formulation is widely used in machine learning and deep learning, where statistical interpretation of individual coefficients is less critical than predictive performance.

The softmax function ensures all class probabilities sum to 1:

$$P(Y = k|X = \mathbf{x}) = \frac{e^{\beta_{\mathbf{k}}^T \mathbf{x}}}{\sum_{l=1}^{K} \mathbf{e}^{\beta_{\mathbf{l}}^T \mathbf{x}}}$$

Note that this provides an **overparametrization** of the model, as it includes more parameters than strictly needed.

The Softmax regressor is optimized using a similar loss function to the Multinomial and Logistic regressors.

Softmax Decision Boundaries

The core decision rule for a Softmax regressor is to assign each observation to the class with the highest predicted probability. Unlike multinomial logistic regression, which uses a baseline class, Softmax treats all classes symmetrically.

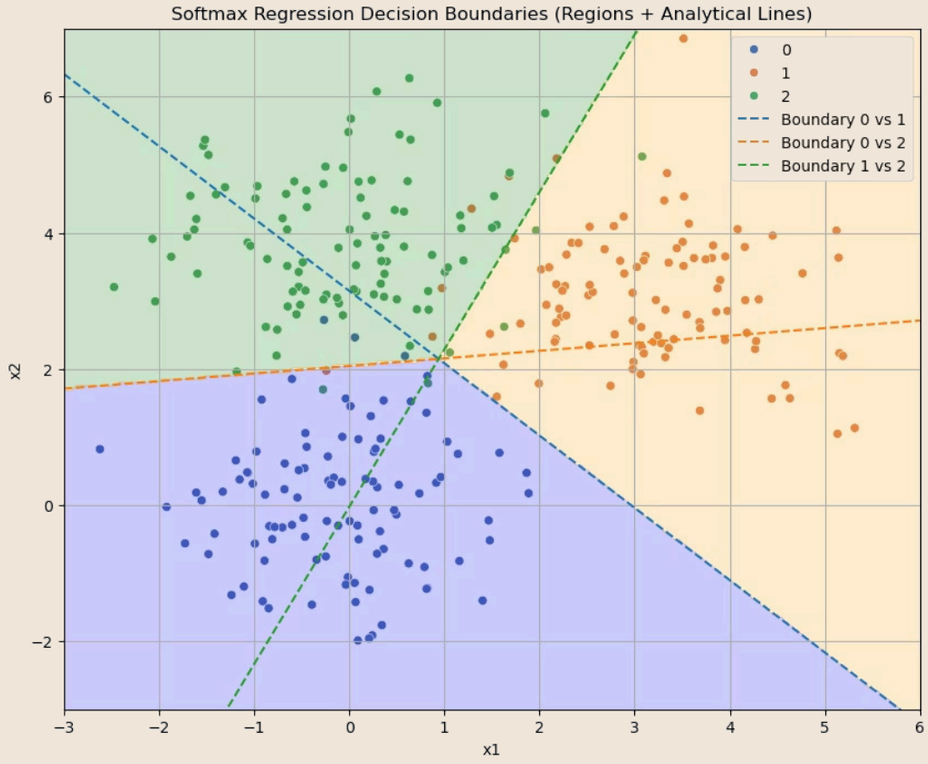
The **decision boundary** between any two classes i and j is formed where their probabilities are equal:

$$P(Y = i|\mathbf{x}) = P(Y = j|\mathbf{x})$$

Through mathematical simplification, this condition directly translates to the equality of their linear score functions:

$$\beta_i^T \mathbf{x} = \beta_j^T \mathbf{x} \Rightarrow (\beta_i - \beta_j)^T \mathbf{x} = 0$$

Geometrically, this equation defines a **hyperplane** in the feature space. In two dimensions, this is a straight line, and in three dimensions, it's a flat plane.



This means that Softmax regression effectively partitions the feature space into distinct regions, with each region corresponding to the class that is most likely according to the model. These partitions are always separated by linear boundaries.

Linear Boundaries

Softmax regression generates **linear decision boundaries** between classes, similar to binary logistic regression, but extended to multiple classes.

Score Equality

Each boundary is defined by the equality of two linear functions, $\beta_i^T \mathbf{x}$ and $\beta_j^T \mathbf{x}$, for any pair of classes.

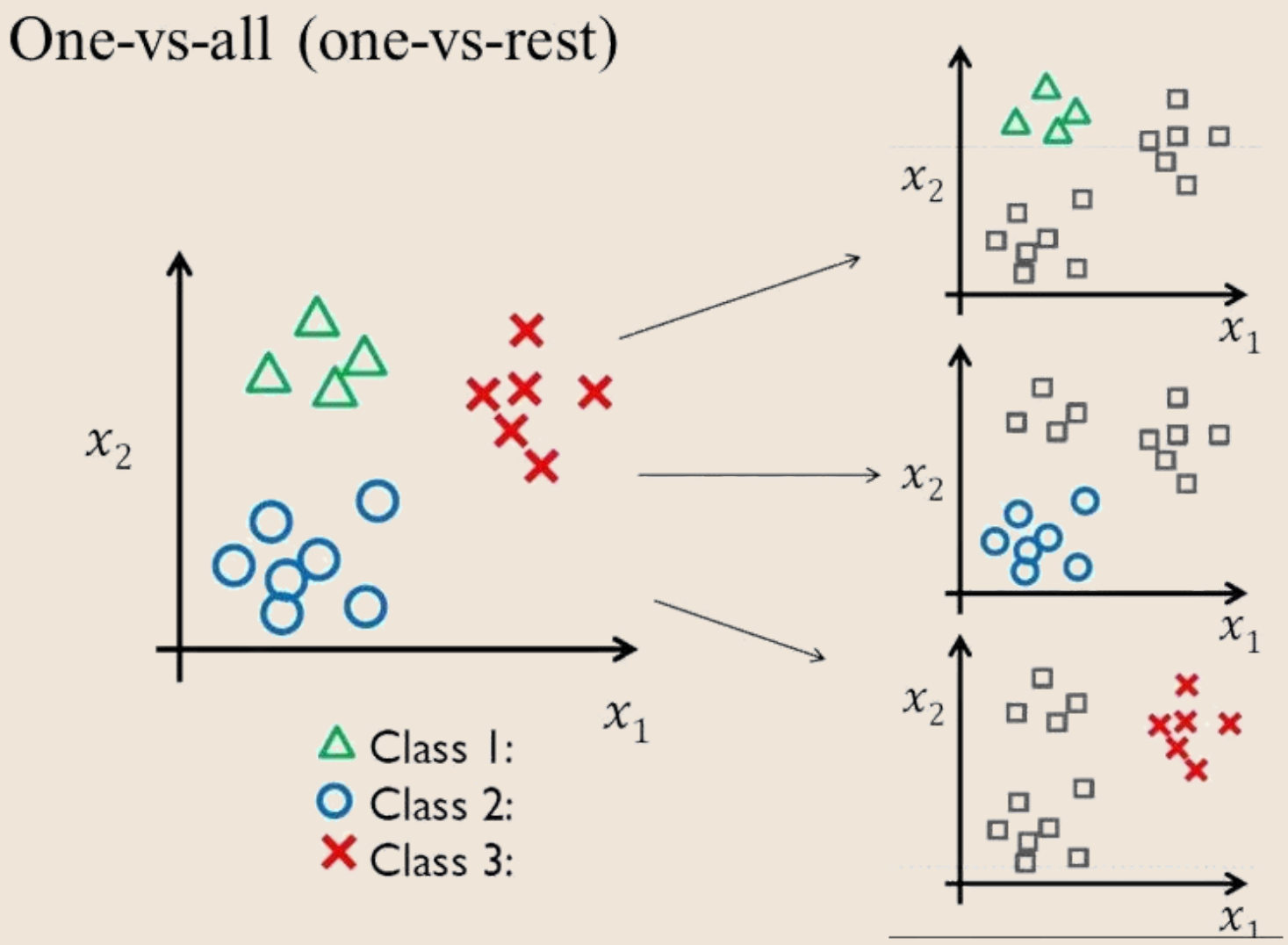
Space Partitioning

The model systematically divides the feature space into regions, each assigned to the class with the highest probability score.

Coefficient Role

The learned weights β_k for each class determine the precise orientation and position of these separating hyperplanes.

From Binary to Multi-Class: One-vs-Rest (OvR)



One-vs-Rest provides a technique to **turn any binary classification model into a multi-class model**. This is also known as "one vs all".

The OvR approach turns one K-class problem into K separate binary classification problems. To do this, we need a binary classifier (like Logistic Regression) that outputs a confidence score (a probability), which we can use to find the most likely class.

Details

Given a classification task with K classes, the OvR approach works as follows:

1

1. Deconstruct the Problem

We train K independent binary classifiers, denoted as $h_k(\mathbf{x})$, for $k \in \{1, \ldots, K\}$. Each classifier h_k is trained to distinguish class k from all other classes (the "rest").

Specifically, for each classifier h_k, the training set is relabeled:

- **Positive examples**: Instances belonging to class k.
- **Negative examples**: Instances belonging to any class $j \neq k$.

For instance:

- Classifier h_1: Distinguishes Class 1 from $\{\text{Class } 2, \ldots, \text{Class } K\}$
- Classifier h_2: Distinguishes Class 2 from $\{\text{Class } 1, \text{Class } 3, \ldots, \text{Class } K\}$
- ...
- Classifier h_K: Distinguishes Class K from $\{\text{Class } 1, \ldots, \text{Class } K-1\}$

2

2. Classify a New Instance

To classify a new input instance $\mathbf{x}$, it is fed to all K trained binary classifiers. Each classifier h_k outputs a confidence score, typically a probability $P(y = k|\mathbf{x})$, indicating the likelihood that $\mathbf{x}$ belongs to class k.

Mathematically, for each class k, we obtain:

$$s_k(\mathbf{x}) = P(y = k|\mathbf{x})$$

Where $s_k(\mathbf{x})$ represents the confidence score for class k.

3

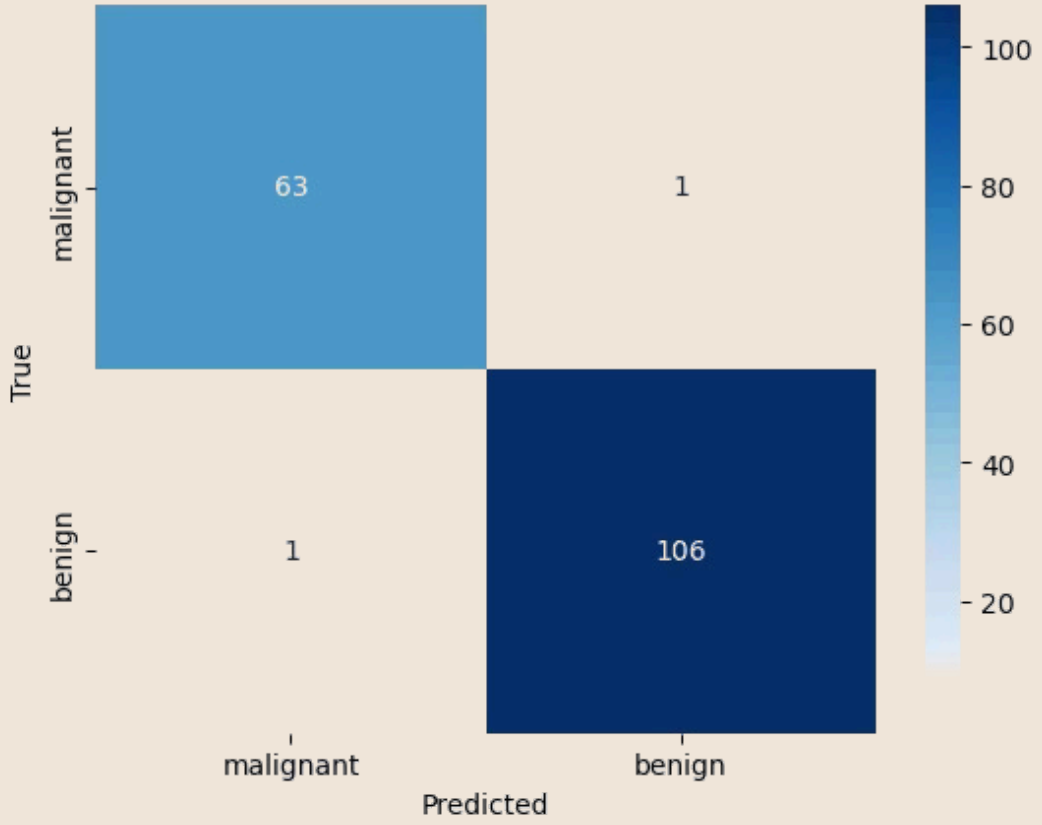
3. Select the Predicted Class

The final predicted class, $\hat{y}$, is determined by selecting the class whose corresponding classifier yielded the highest confidence score among all K classifiers.

$$\hat{y} = \underset{k \in \{1, \ldots, K\}}{\operatorname{argmax}} \; s_k(\mathbf{x})$$

This decision rule assigns the instance $\mathbf{x}$ to the class k for which its binary classifier h_k is most confident.

Multiclass Logistic Regression in Python



Conclusions and Next Steps

We Have Explored:

- Multinomial logistic regression
- Geometrical interpretation of the coefficients of a logistic regressor
- Softmax regressor
- Geometrical interpretation of the coefficients of a softmax regressor
- One vs rest

In the next lectures, we will look at the Naive Bayes classifier

References

- Chapter 4 of [1]
- https://medium.com/@b.terryjack/tips-and-tricks-for-multi-class-classification-c184ae1c8ffc

[1] James, Gareth Gareth Michael. An introduction to statistical learning: with applications in Python, 2023.https://www.statlearning.com